THIRD ANNUAL

2003 Microsoft® STRATEGIC ARCHITECT FORUM

for Customers

Information and Application Architecture for the Service-Oriented Enterprise: Metropolis—Part 2

Pat Helland
Architect
.NET Enterprise Architecture Team
Microsoft Corporation

Dave Campbell
Product Unit Manager
SQL
Microsoft Corporation

Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - **Introduction to Part 2**
  - Considering Data and Messaging in Services
  - Thoughts on Business Process
  - Conclusion

Summary of the Previous Session

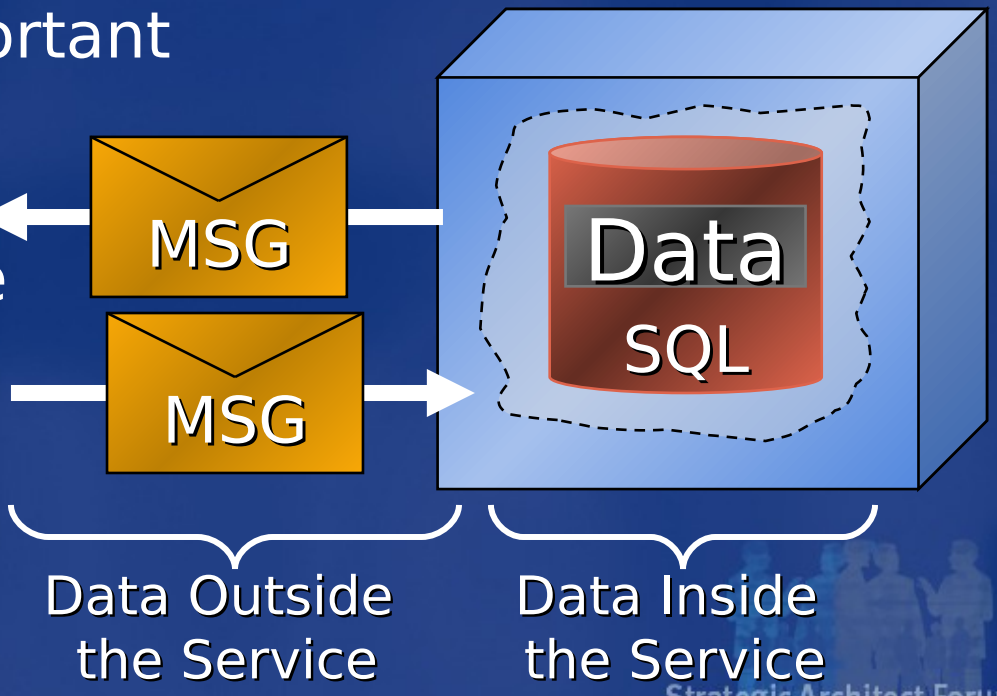
The Metropolis Analogy

Practical Advice for Building Your Services *Now* !

Avoiding Ambiguity in Messages	What to consider when writing messages to ens they are understood by your partner
Services for the Enterprise	Special considerations for using services for enterprise-class applications
Message Delivery in the Mean, Cruel Worlc	Connecting apps (services) with messages has many pitfalls and failure windows… What to do?
Implementing Your Business Service	Messages, data, and transactions… How can we make an enterprise-class service work?
Embracing and Extending Your Apps	I've got a LOT of existing applications… How ca they be tied into the services game?

Data Inside and Outside the Service

- Data Is Different Inside from Outside
- Outside the Business Service
  - Passed in Messages
  - Understood by Sender and Receiver
  - Independent Schema Definition Important
  - Extensibility Important
- Inside the Business Service
  - Private to Service
  - Encapsulated by Service Code



Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - **Considering Data and Messaging in Services**
  - Thoughts on Business Process
  - Conclusion

Considering Data and Messaging in Services

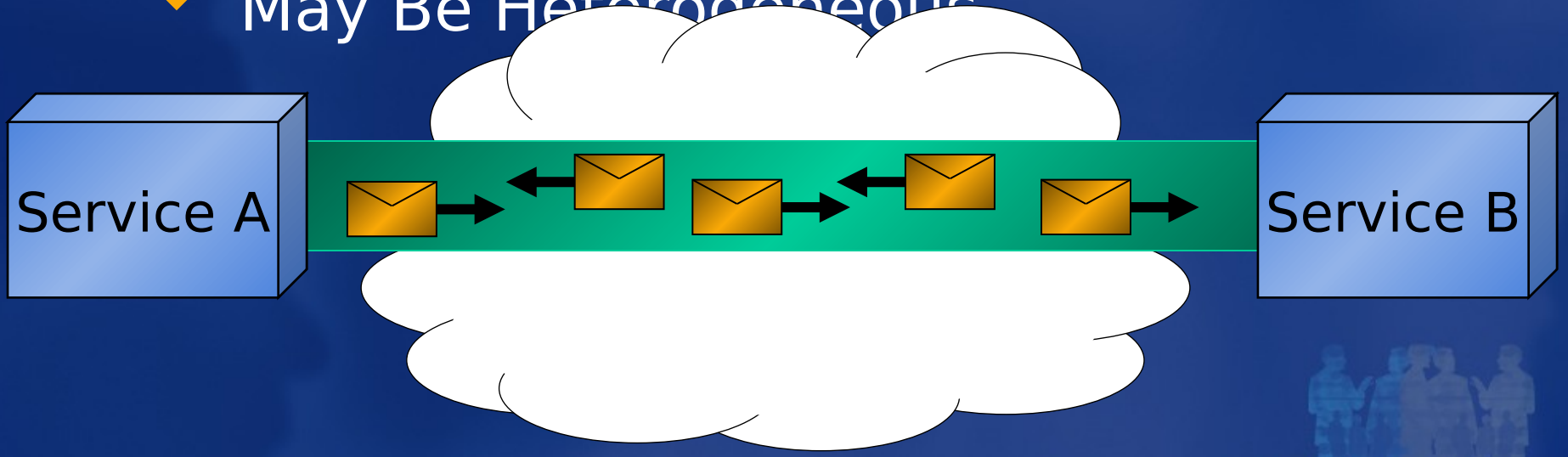
The Space Between the Services	What goes in the messages being passed betwe services? What must be considered?
The Schema Between the Services	What about the different understandings of data Why is XML such a big deal?
Service Agents and Service Masters	How is data used inside a service? What are the usages patterns that can be leveraged?
Ownership of Data in the Enterprise	If data is changeable, only one service should b in charge... How should this be done?
Representations of Data	XML, SQL, Objects! When do I use what? Why so many choices?
Tying It All Together!	Reiterating the roles for XML, Objects, and SQL

Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - **Considering Data and Messaging in Services**
    - **The Space Between the Services**
    - Schema in the Space Between the Services
    - Service Masters and Service Agents
    - Ownership of Data in the Enterprise
    - Representations of Data
    - Tying It All Together
  - Thoughts on Business Process
  - Conclusion

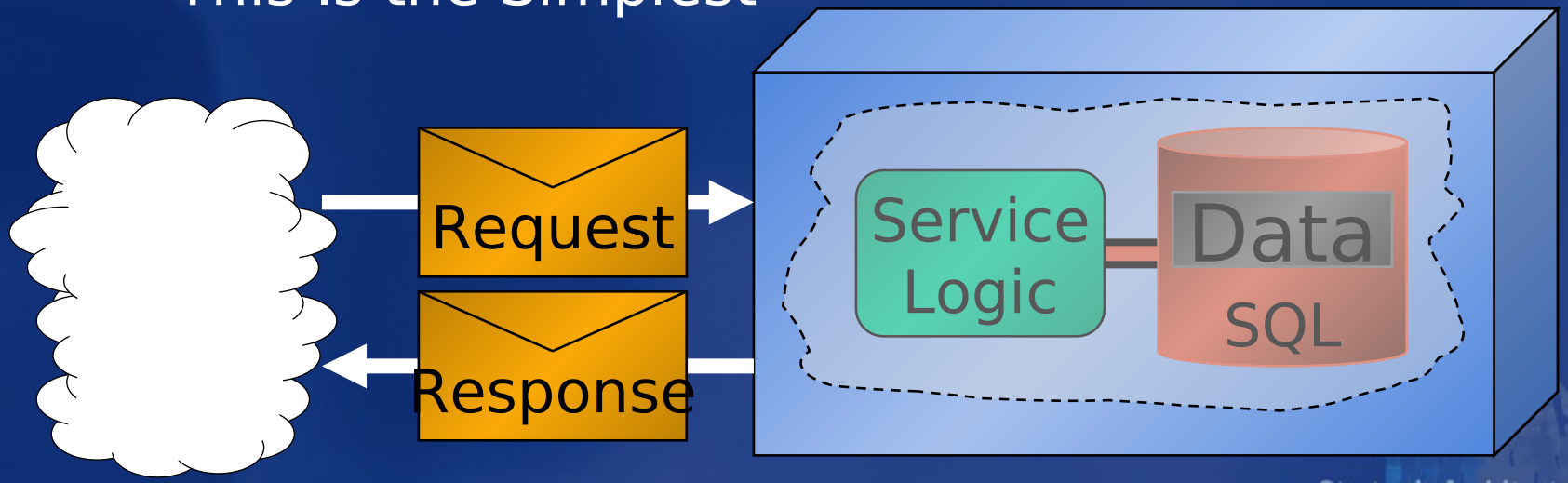
Services Communicate with Messages

- Services Communicate with Messages
  - Nothing Else
- No Other Knowledge About Partner
  - May Be Heterogeneous



Service Requests and Services Responses

- Service Requests
  - Ask for Work from the Service
- Service Responses
  - Carry the Response to the Request
- May Be Other Patterns
  - This Is the Simplest



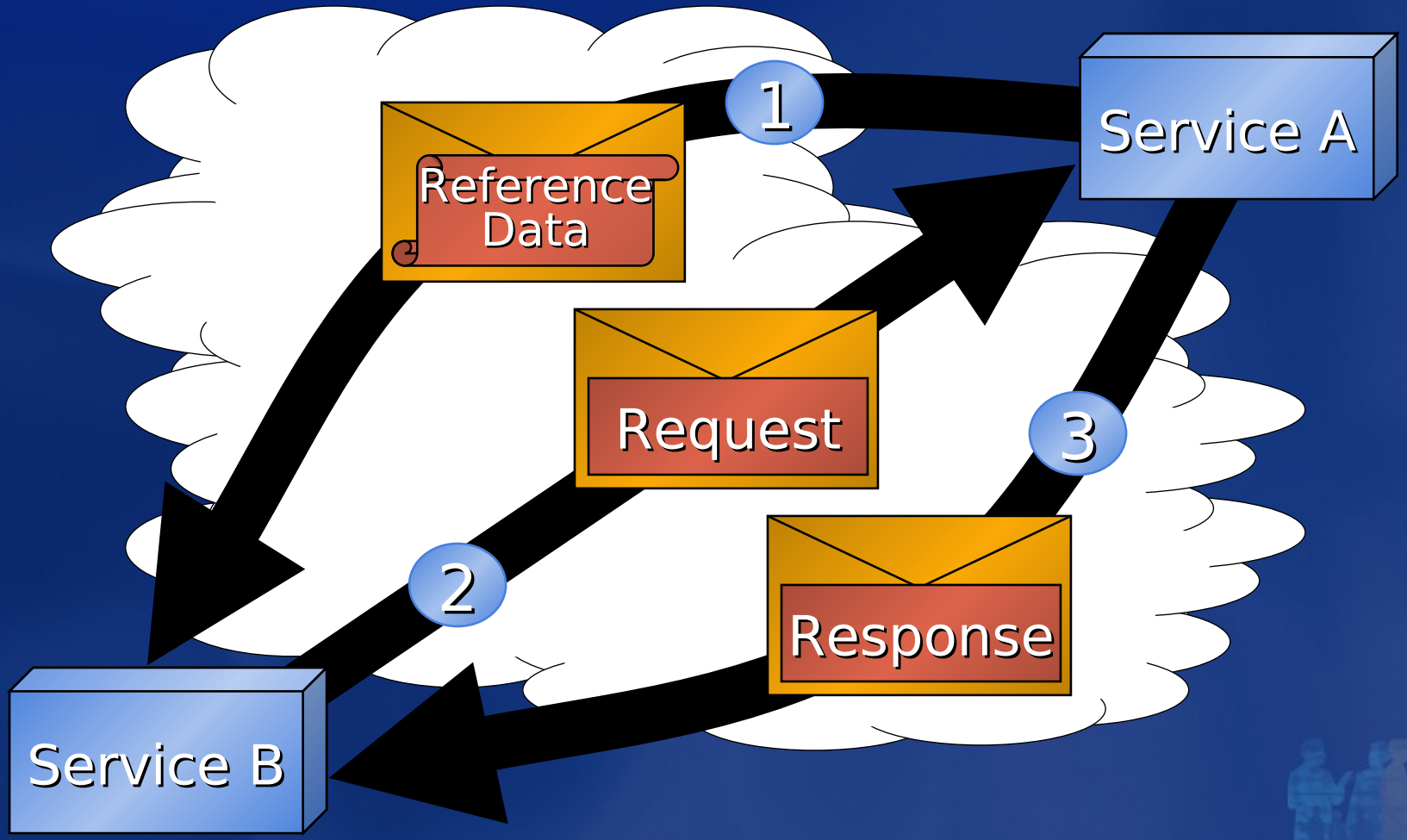
Messages Carry Requests, Responses, and Reference Data

- Services Interact with Messages
  - Can Have Useful Discussions About Just Messaging
- Messages May Carry Reference Data
  - Data Used to Fill out Requests/Responses
- Messages May Carry Requests
  - Retry-able Descriptions of Work
- Messages May Carry Responses
  - Answers to Requests (Saying What Happened)
- May Use Request/Response to Move Reference Data
  - OK to Push or Pull Reference Data

Publication of Reference Data

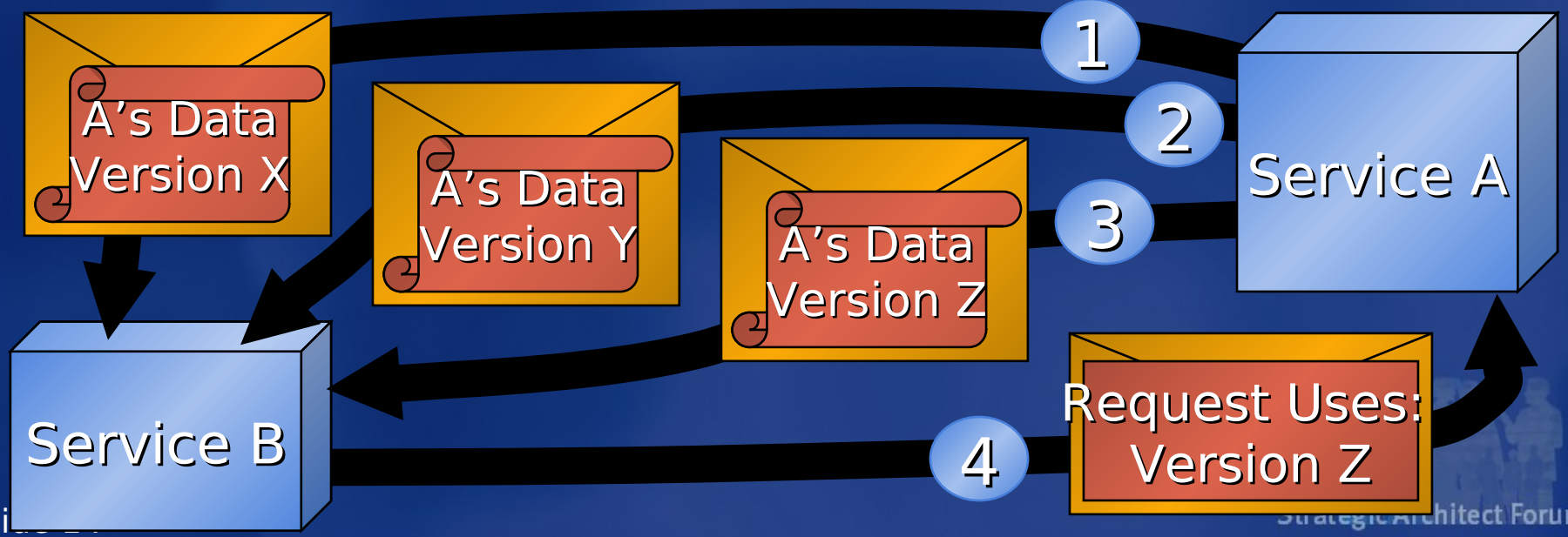
- Publish Reference Data However You Want
  - Push
    - E-Mail, HTTP, IP, TCP/IP, MSMQ, MQ Series, CD-ROM, DVD, Carrier Pigeon, etc.
  - Pull
    - FTP, HTTP, E-Mail Requesting CD-ROM, Psychic Hotline, etc.
- Data Sent from Publisher to Subscriber
  - Periodically Updated
  - Always Versioned
- Just Like Department Store Catalog

Request/Response and Reference Data



Versioning of Reference Data

- Reference Data Has Identity
  - Describes the Publisher and Purpose
- Reference Data Gets Updated Periodically
  - Usually at Defined Interval
- Need a Version Scheme
  - Describes Which Version You Are Using



Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - **Considering Data and Messaging in Services**
    - The Space Between the Services
    - **Schema in the Space Between the Services**
    - Service Masters and Service Agents
    - Ownership of Data in the Enterprise
    - Representations of Data
    - Tying It All Together
  - Thoughts on Business Process
  - Conclusion

Documents, Messages, Independence, Extensibility, and XML

- XML Grew Up from the Document World
  - SGML → HTML → XML
- Core Tenets of XML Are:
  - Independence of Definition
    - Anyone Can Add Stuff
    - A Document May Combine Different Stuff
  - Extensibility
    - Sure… Add Stuff
    - I May or May Not Care

Infosets, XML Schema, and PSVI

- XML Infoset
  - Semantics of XML, Not Syntax
  - Tree: Parents, Children, Elements, and Attributes
  - Allows (Encourages) Schema
  - Any Representation OK
- XML Schema
  - Datatype Library and Schema Definition
  - Composed Schema Uniquely Identified (URI)
- PSVI: Post-Schema-Validation Infoset
  - Infoset After Validation Against Schema
  - Can Leverage Schema Knowledge

Immutability, Infosets, and Messages

- Messages Must Be Immutable
  - Retries Must Not See Differences…
- Immutable Infosets in Messages
  - Once It's Written, It's Written!

When It's Outside,
It's Immutable!

Extensibility and XML Schema

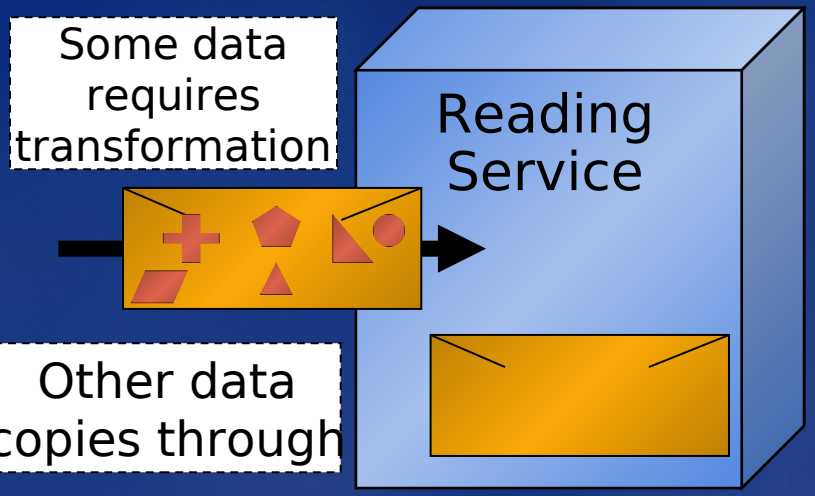
- Each Message Should Have a Schema
  - Describes the Contents
  - Schema Defined with Unique ID
    - Typically a URI
- Must Be Able to Find the Schema from the ID
  - Commonly Done with URL
- We Recommend Using XML Schema
  - Well-Established Standard
  - XML Schema Supports Extensibility
    - Allows Nesting of Schemas
  - Can Program to Extract Understood Parts of Schema

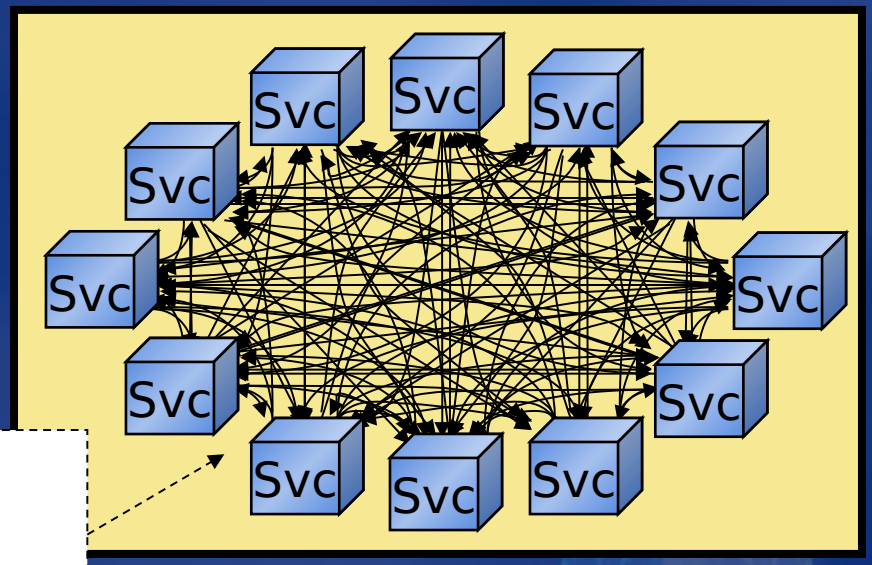
Climbing the Tower of Babel

- Lots of Independently Designed Systems
  - Many Differences at Many Levels
- Moving Up the Standardization Stack
  - Now Only Messages and Contracts Matter
    - Contracts Define Message Sequences
- Huge Challenge to Rationalize
  - Semantics of Business Data Very Hard
  - Semantics of Business Operations Very Hard
- Schema Rationalization Ongoing
  - Working Our Way Up the Stack

Reader-Makes-Right and N-Squared Combinatorics

- Reader-Makes-Right
  - Receiver Fixes Message
  - Based on Both Schemas
  - Best Possible Mapping
- N-Squared Combinatorics
  - Everybody Knows Everybody
  - N*(N-1) Mappings
  - Gets Big Fast

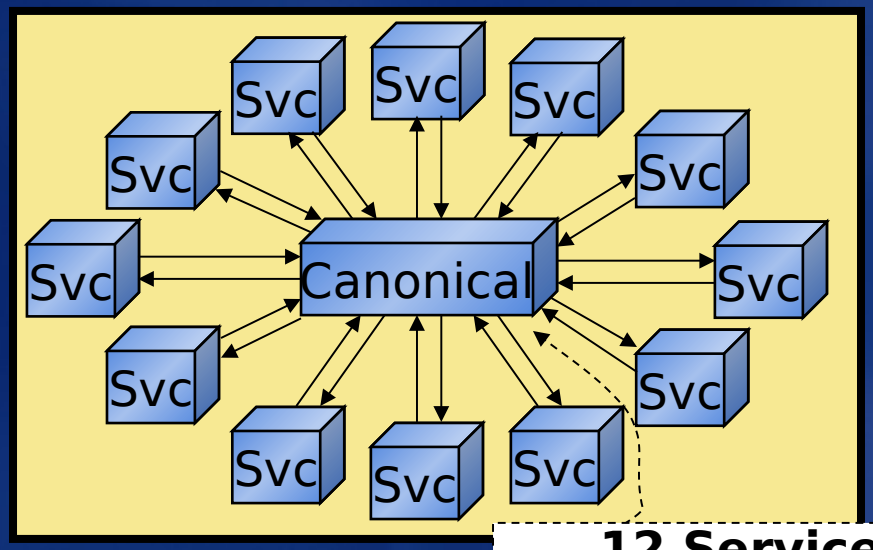




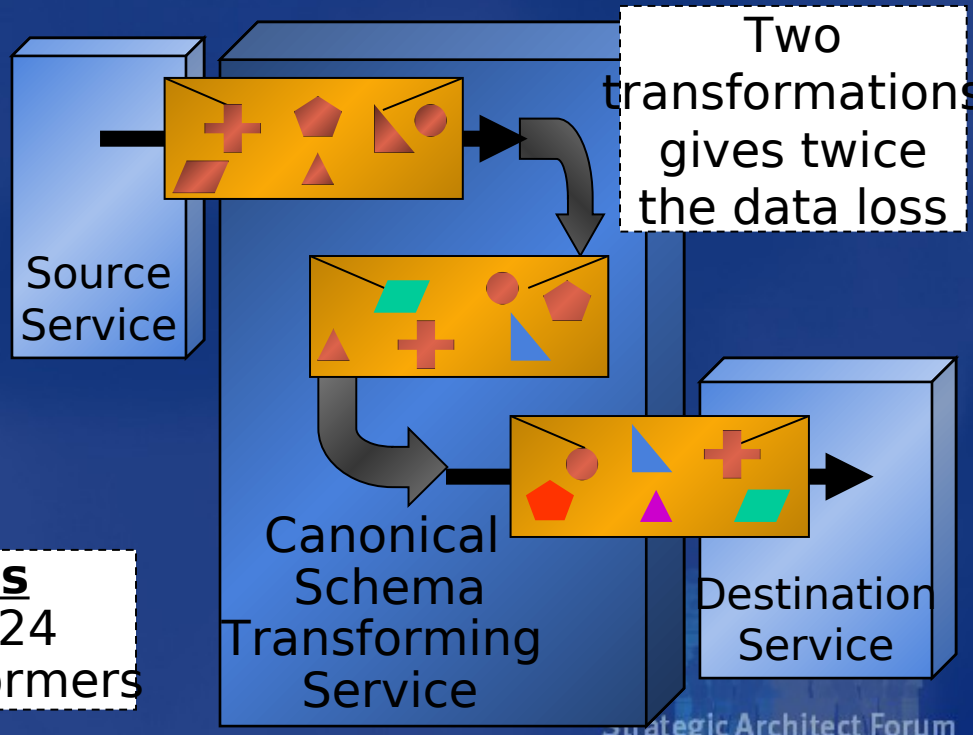
12 Services
12 * 11 = 132
message transformers

Canonical Schema and Doubly Lossy Transformations

- *Canonical Schema* Is a Standard Representation
  - Map Everything to the Canonical and Back Out
  - Far Fewer Transformers Required
- Using a Canonical Schema Is *Double-Lossy*
  - Info Loss Going in and Info Loss Going Out



12 Services
2 * N; 2*12=24
message transformers

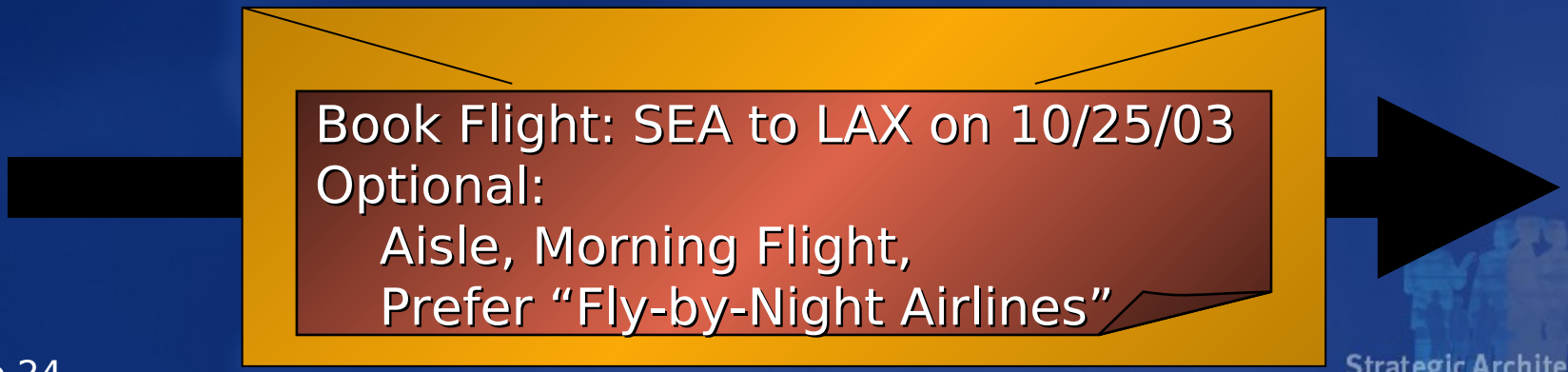


The Typical Approach to Integrating Schema

- Classic Solution:
  - Implement Canonical Schema
    - Practical to Build the Transformers
    - 2 * N (for N Different Schemas)
  - Specialize When Needed
    - If Doubly Lossy Hurts, Then Reader-Makes-Right
    - Specialize Only Where Important
- Hybrid Trades Work for Fidelity

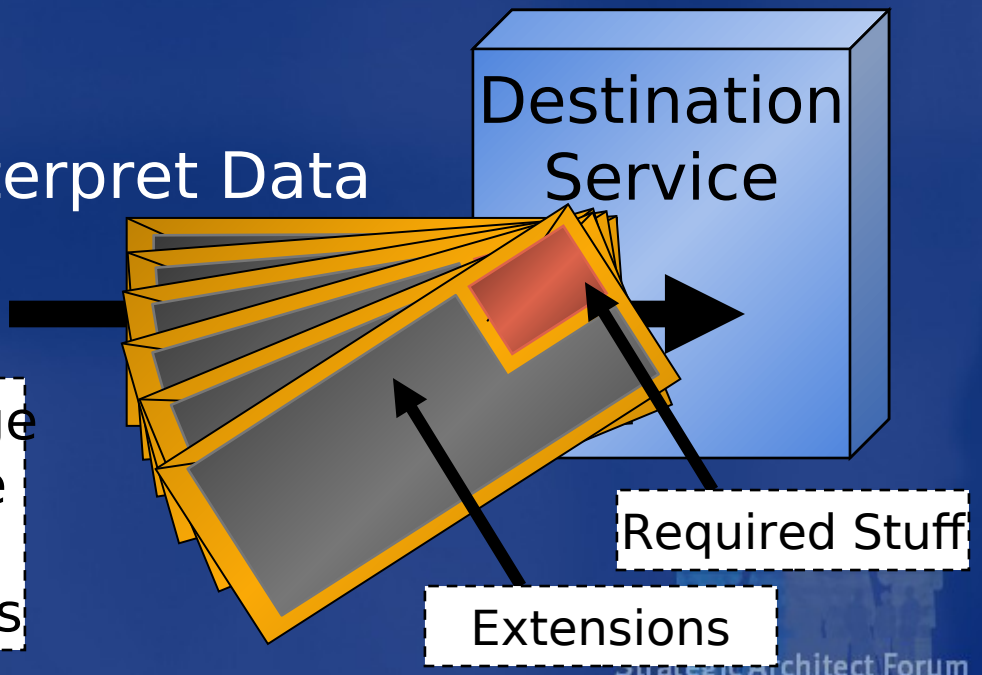
Early Binding vs. Late Binding

- Early Binding
  - All Details Are Shared
  - Classic Method Calls Are Early Bound
    - Everything Known in Advance
- Late Binding
  - Minimum Subset Known in Common
  - Sender Can Add Interesting Stuff
  - Receiver Looks for Added Stuff



Advantages of and Challenges with Late Binding

- Advantages:
  - Flexibility
    - Sender and Receiver Evolve Independently
  - Auditing
    - Record Extensions Even If You Don't Understand
- Challenges:
  - Hard to Code to Interpret Data



The incoming message can pivot around the required data and add lots of extensions

Thoughts on Late Binding

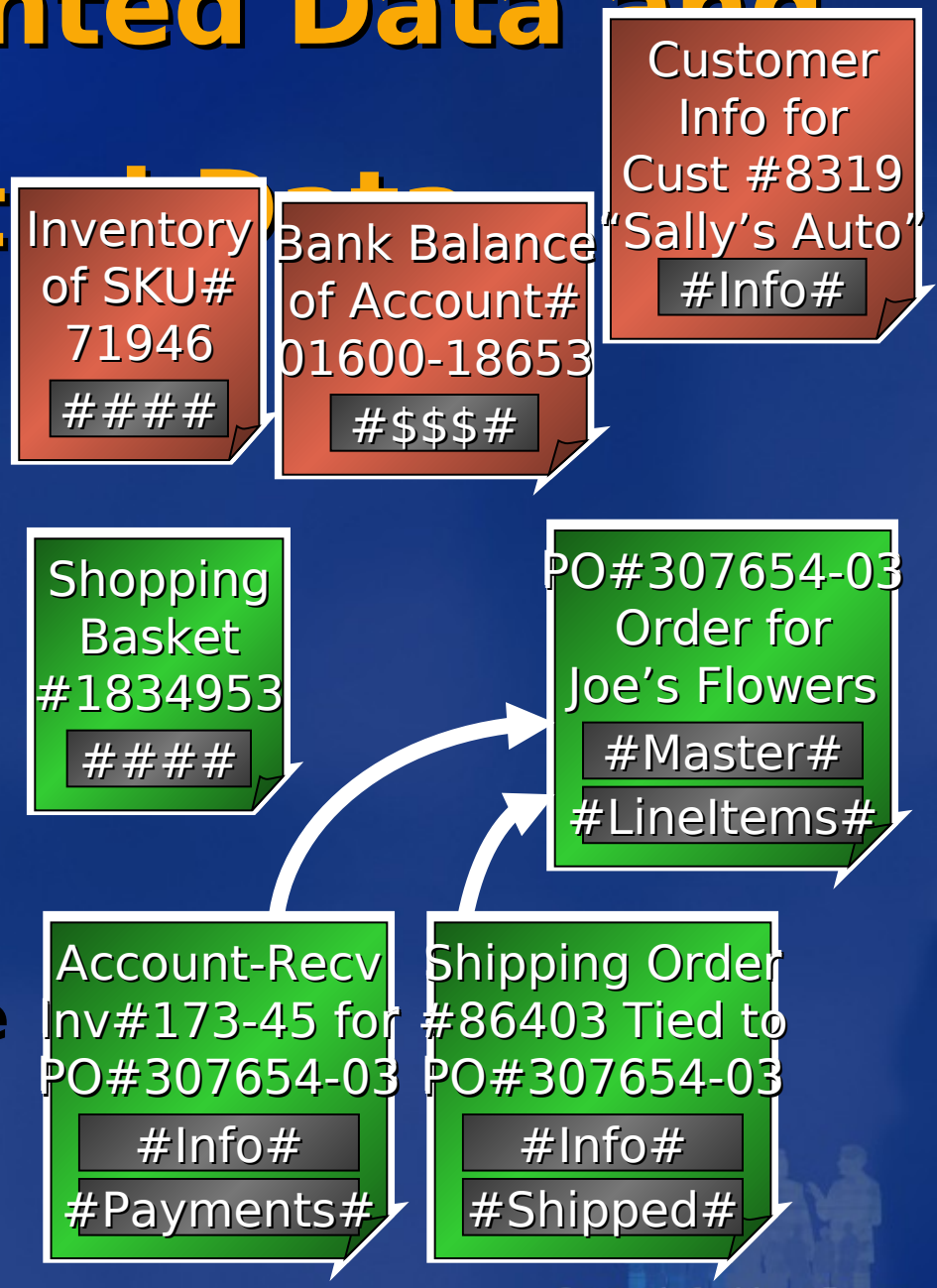
- Parameters
  - What Is Minimum Required?
  - What Can Be Optional?
- Judgments About Minimum
  - Try to Codify What Humans Know Is Minimal
- Different Attitude
  - Maximize Independence
  - Record All Incoming Information
    - For Auditing (If No Other Reason)
  - Try to Use Optional Data to Add Value

Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - **Considering Data and Messaging in Services**
    - The Space Between the Services
    - Schema in the Space Between the Services
    - **Service Masters and Service Agents**
    - Ownership of Data in the Enterprise
    - Representations of Data
    - Tying It All Together
  - Thoughts on Business Process
  - Conclusion

Resource-Oriented Data and Activity-Oriented Data

- Resource-Oriented Data
  - Lives Longer Than One Long-Running Operation
  - Changed by Long-Running Operations
- Activity-Oriented Data
  - Gets Modified by Long-Running Operations



These classes of data have different characteristics.

We're gonna examine some of the consequences of these differences...

Activity-Oriented Data Retirement

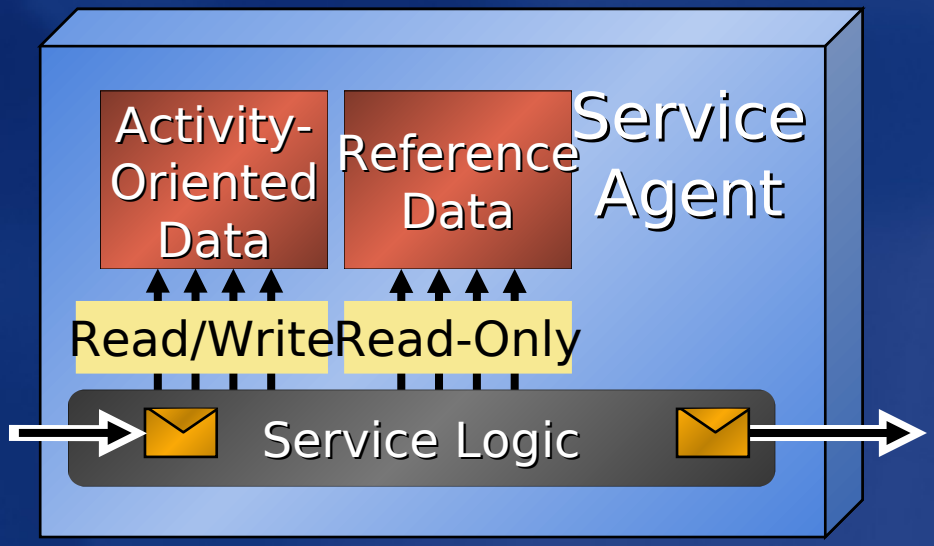
- Long-Running Operations Take Time
  - Frequently Weeks or Months
  - Eventually, They Complete
- Activity-Oriented Data Retires
  - It Becomes Read-Only
  - It Is Referenced Less and Less Frequently
    - It May Get Archived to Tape
  - Rarely, It Is Deleted
    - New Regulations Will Make This Very Rare

Resource-Oriented Data Retirement

- Resource-Oriented Data May Live a Long Time
  - SKUs, Accounts, Employees, Customers, etc.
  - These May Live Years
- Sometimes These Leave
  - Discontinue SKU, Fire Employee, etc
- Some Resources Have Bounded Life
  - Seat Assignments on an Airplane Flight
  - Hotel Occupancy on a Particular Night
- Resource-Oriented Data May Retire
  - It Enters a Read-Only Status
  - It May Get Archived to Tape

Service Agents

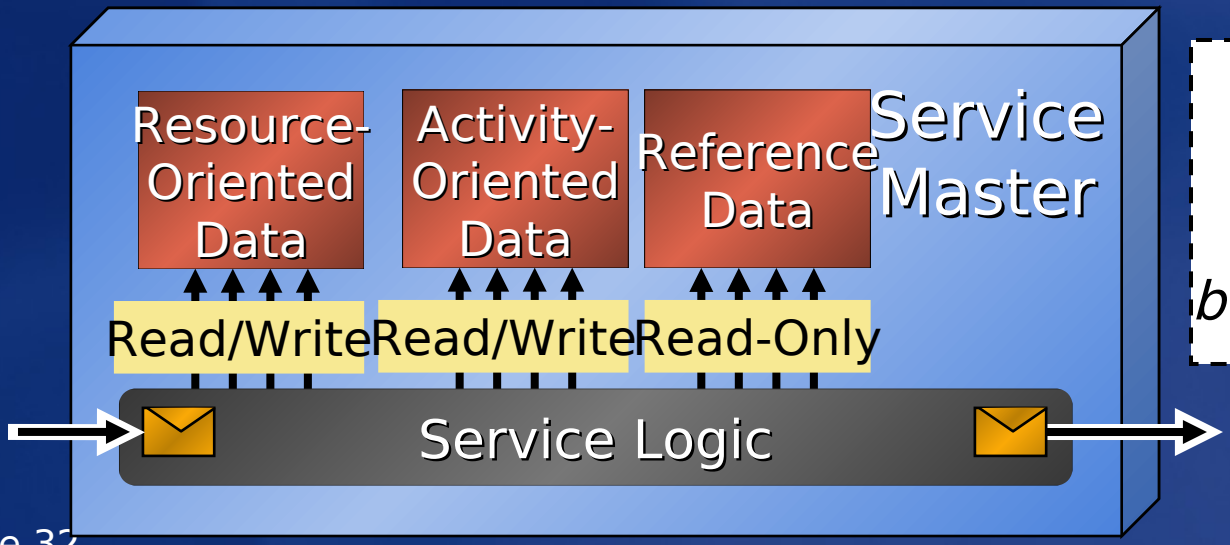
- A Service Agent Is a Special Service
  - Manages Activity-Oriented Data
  - Lives for a Single Long-Running Operation
  - Does Not Use Resource-Oriented Data
- Uses Only:
  - Activity-Oriented Data
  - Requests/Responses (Incoming and Outgoing)
  - Reference Data



This is very similar (but not identical) to what used to be called "Emissaries."

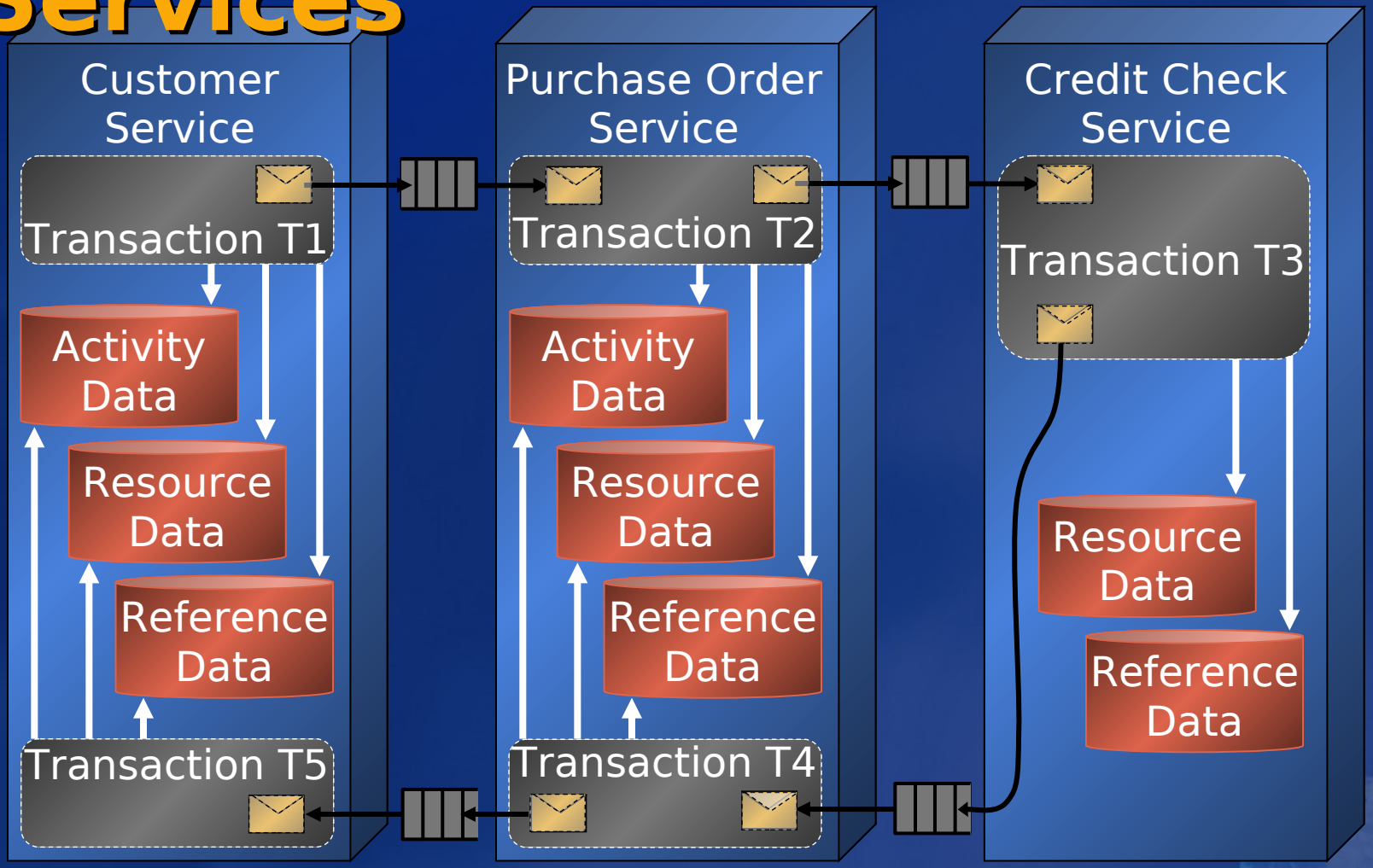
Service Masters

- A Service Master Is a Special Service
  - Manages Resource and Activity Data
  - Lives Longer Than a Long-Running Operation
- Uses All Kinds of Data:
  - Reference-Oriented Data
  - Activity-Oriented Data
  - Requests/Responses (Incoming and Outgoing)
  - Reference Data



This is very similar (but not identical) to what used to be called "Fiefdoms."

Activity-Oriented Data and Work Across Multiple Services

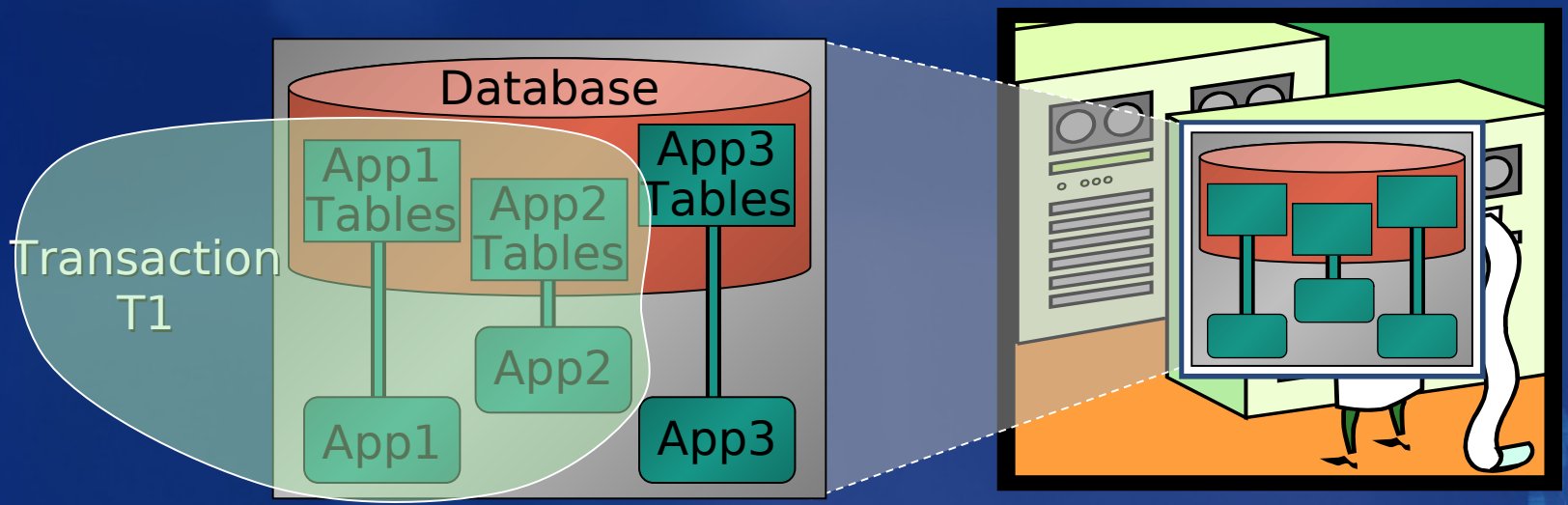


Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - **Considering Data and Messaging in Services**
    - The Space Between the Services
    - Schema in the Space Between the Services
    - Service Masters and Service Agents
    - **Ownership of Data in the Enterprise**
    - Representations of Data
    - Tying It All Together
  - Thoughts on Business Process
  - Conclusion

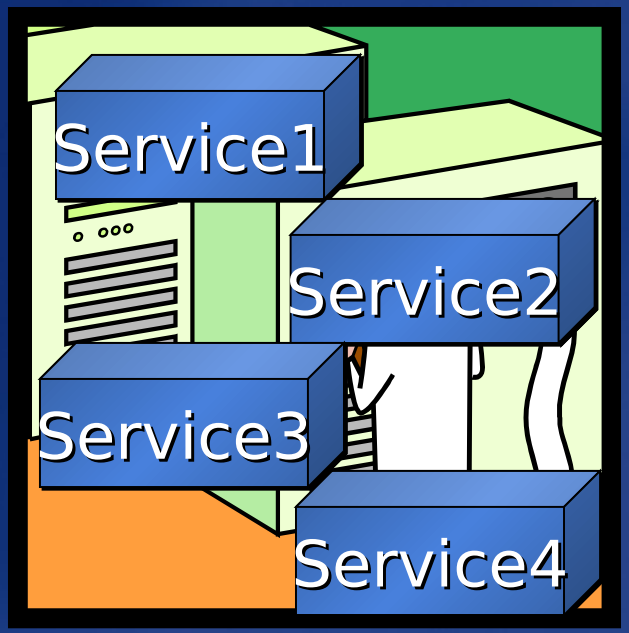
Mainframes and Monoliths

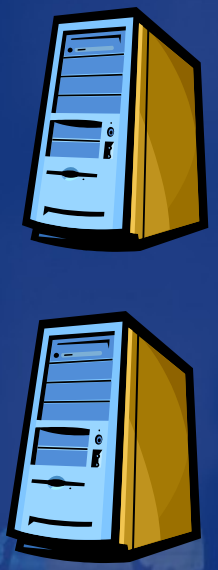
- Multiple Apps in One Mainframe
  - One Database with Data for Lots of Apps
    - Usually Worked on Different Tables
- Different Apps Could Share a Transaction
  - Operations Could Be Atomic
- Could See Other Apps' Data



The Fracturing of the Monoliths

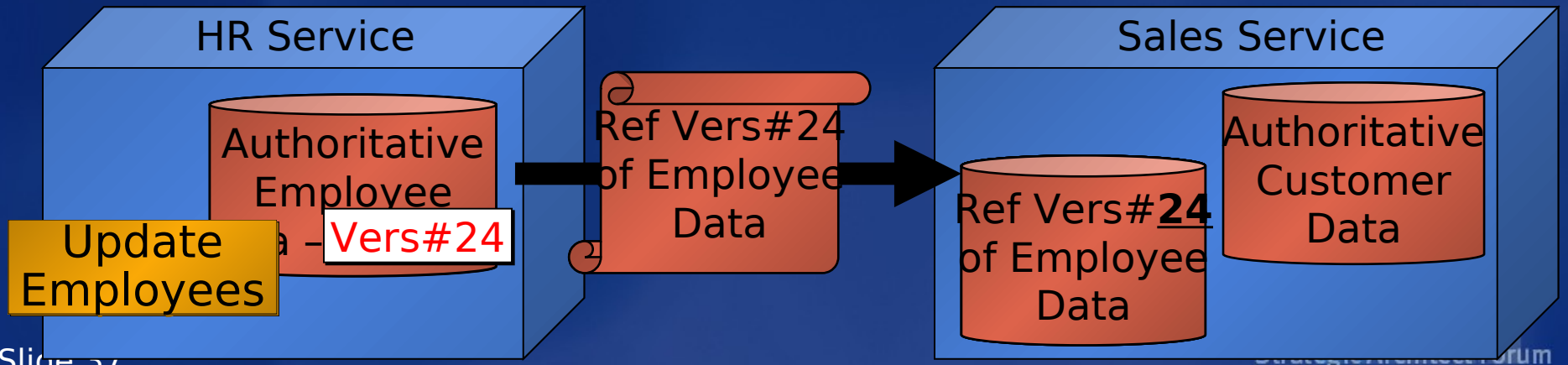
- Mainframe Apps Are Moving!
  - Different Machines → Different Transactions
- It Is Harder to Share Data!
  - Can't Just Read the Latest and Greatest Info
  - Must Maintain *Copies* on Distant Machines…





When There's Data Needed by Many

- Data May Be Needed by Many Services
  - Customers, Employees, Parts, etc.
- Each Piece of Data Needs an Owner
  - Only the Owner May Change It
- Owner Publishes Changes to Others
  - Others Receive Updates and Cache Versions

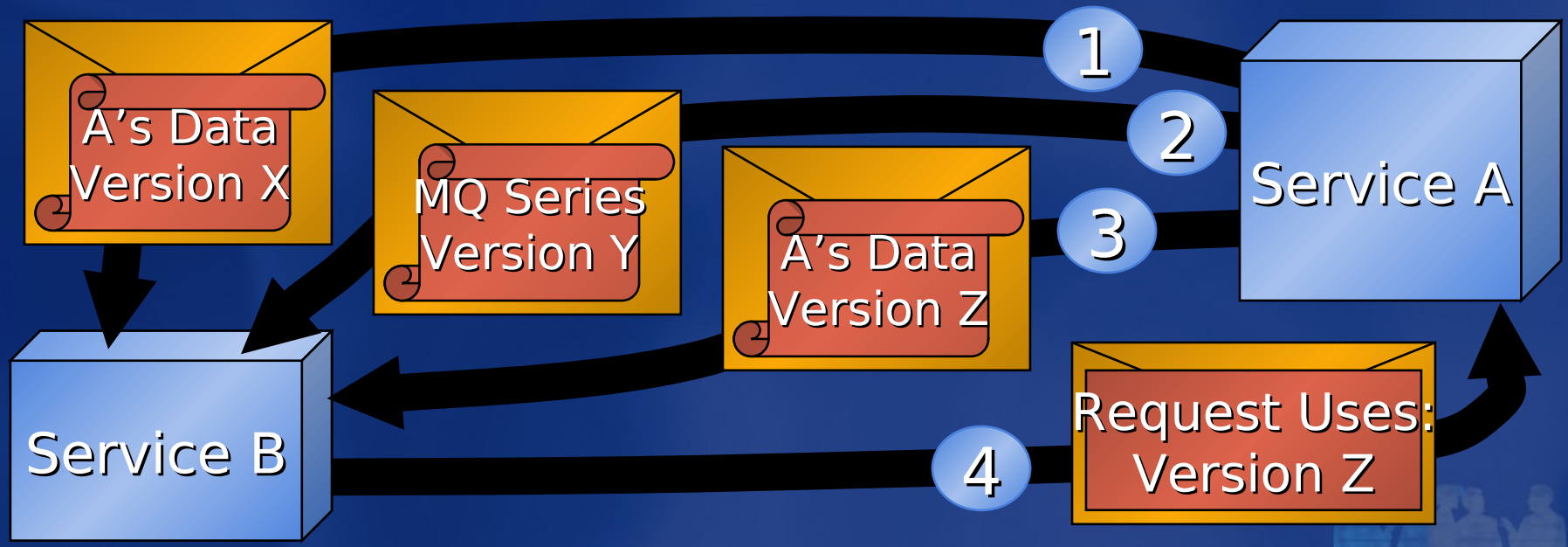


Building a Single Owner for the Data

- Any Piece of Data Has a Single Owner
  - It Has a Private View of the Data
  - It Publishes a Public View of the Data
- You Cannot Have Multiple Owners!
  - Confusion Will Reign…
- Partitioning Is Just Fine!
  - You May Partition by Topic and App
  - You May Partition by Region
  - Each Individual Piece of Data Has One Owner

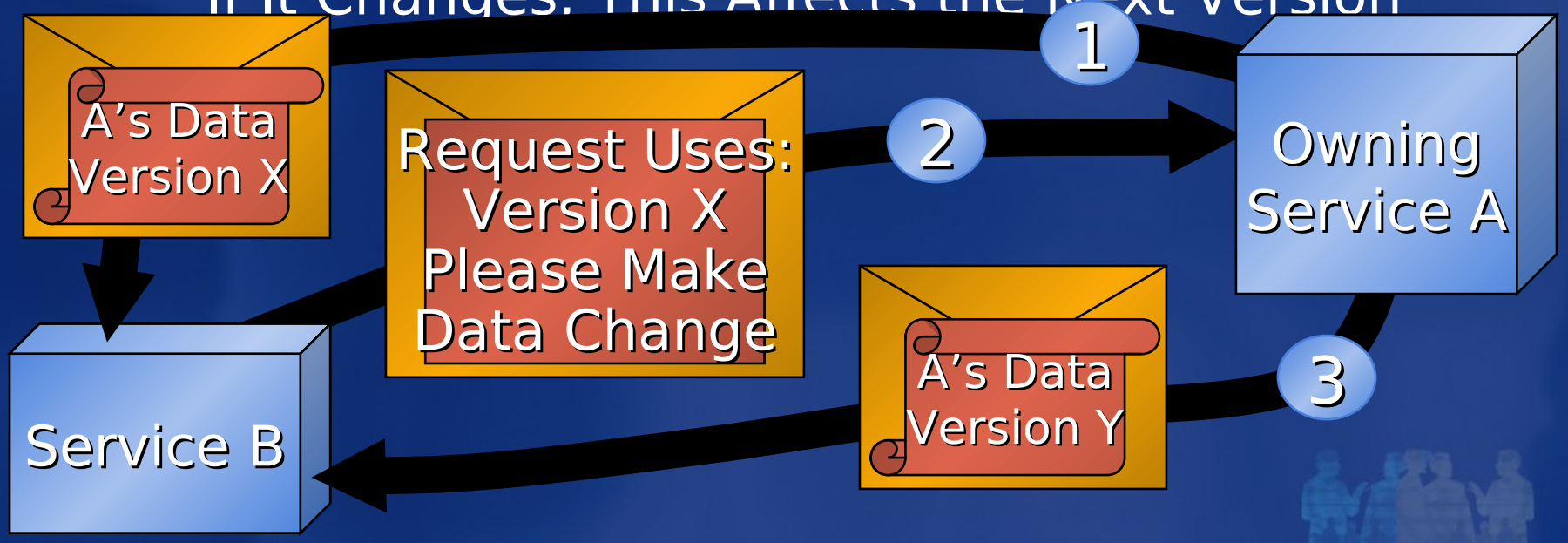
Pushing out Versioned Replicas

- The Owner of Data Periodically Publishes
  - Using Whatever Messaging Technique It Wants
- Publications Are Always Versioned
  - The Version Numbers Increase



Requesting the Owner Make Changes

- If a Non-Owner Wants a Change It Must Ask for the Change
  - This Is a Request Sent to the Owning Service
  - The Owning Service May Agree to Change the Data
  - If It Changes, This Affects the Next Version



Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - **Considering Data and Messaging in Services**
    - The Space Between the Services
    - Schema in the Space Between the Services
    - Service Masters and Service Agents
    - Ownership of Data in the Enterprise
    - **Representations of Data**
    - Tying It All Together
  - Thoughts on Business Process
  - Conclusion

Kinds of Data

	Stable	Normal-ized	Immutable	Con-current Update
Resource-Oriented Data	Usually Not	Yes	No—Very Volatile	Highly Con-current
Versioned Reference Data	Yes	Maybe	Yes: Each Version Written Once	No Update
Request-Response Data	Yes	Maybe	Yes: Written Once	No Update
Activity-Oriented Data	Yes	Maybe	No (Example: Shopping Basket)	Very Low Con-current

Activity data is rarely concurrently updated; tends to have simple structures

Note that versioned reference data and request/response data never change

The Relational Revolution

- Relational Calculates Values by Relating
  - Set-Oriented Value-Based Operation
- Separation of Data from Logic
  - Allows Lots of Optimizations
- SQL Has Premier Data Management
  - High-Performance
  - Backup, Archival, and Other Management

Bounded and Unbounded Data Representations

- Relational Is Bounded
  - Operations Within the Database
  - Value Comparisons Only Meaningful Inside
  - Tightly Managed Schema
- XML Infoset Is Unbounded
  - Open (Extensible) Schema
    - Contributions to Schema from Who-Knows-Where
  - References (Not Just Values)
    - URIs Known to Be Unique
  - XML Infosets Can Be Interpreted Anywhere

Objects, Components, and Complexity

- Objects Are Great for Software Engineering
  - Combine Logic with Data
- The Essence Is Encapsulation
  - Hide Implementation from Users
- Durable Objects Not Well Established
  - In-Memory Encapsulation Firmly Established
  - OO Databases Not Big Successes
  - Work to Establish Persistent Objects via SQL Store

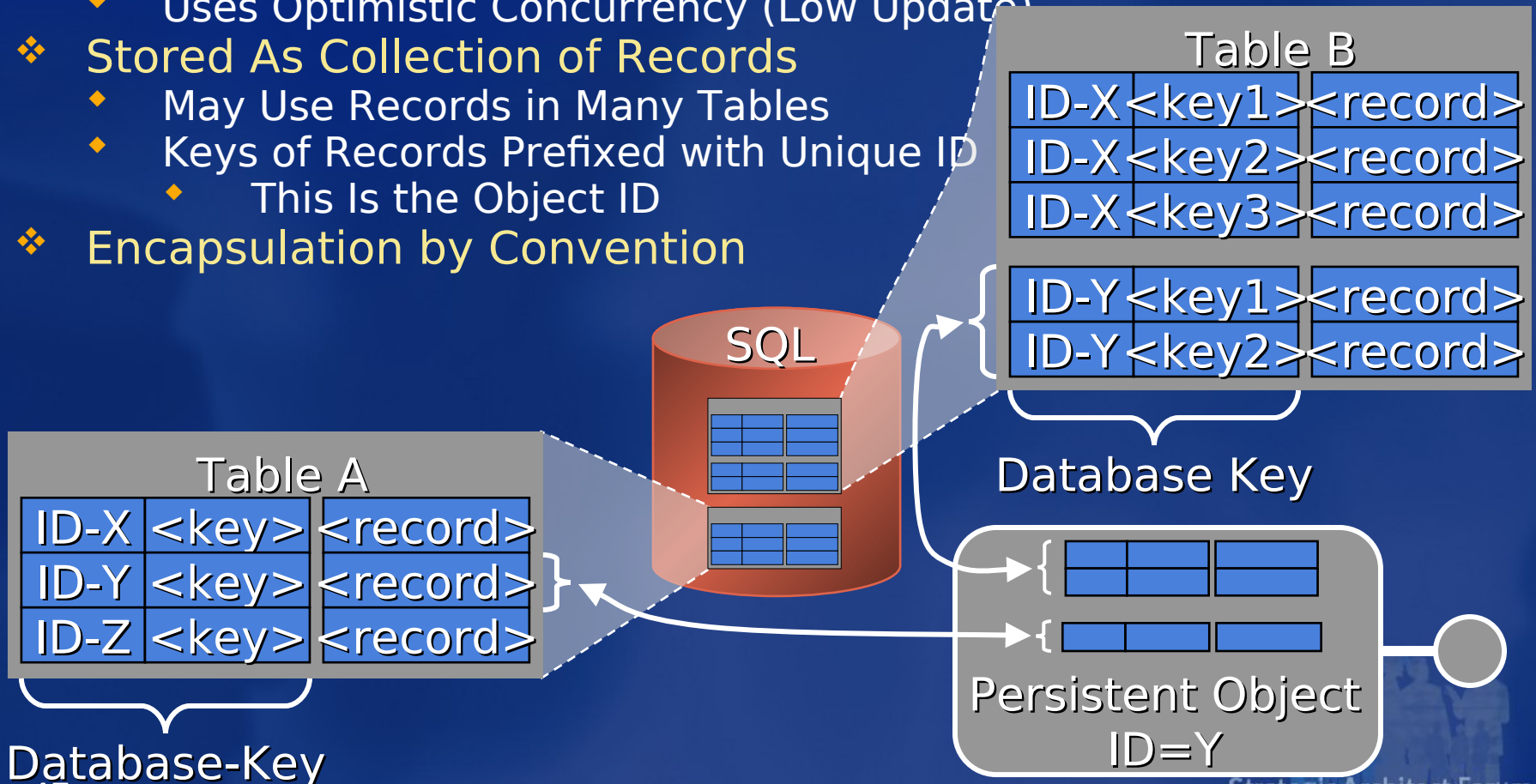
Encapsulating Durable Data

- Objects Encapsulate Data
- SQL and XML Are Naturally Durable
- Components Can Encapsulate Durable Data

Kind of Data	Private to Service?	Needs Encapsulation?	XML?	Concurrency	Object Style	Concurrency Approach
Request Response	No	No: Has Open Schema	Yes	Read-Only	None	None
Reference	No	No: Has Open Schema	Yes	Read-Only	None	None
Activity-Oriented	Yes	Yes	Maybe	Very Low	Object Persistence	Optimistic
Resource-Oriented	Yes	Yes	Maybe	May Be High	COM+ Transactional	Pessimistic (Locking)

Persistent Objects and SQL Storage

- Persistent Objects
  - Encapsulated by Logic
  - Kept in SQL
  - Uses Optimistic Concurrency (Low Update)
- Stored As Collection of Records
  - May Use Records in Many Tables
  - Keys of Records Prefixed with Unique ID
    - This Is the Object ID
- Encapsulation by Convention



Stateless Objects and Resource-Oriented Data

- Resource-Oriented Data Is Changed a Lot
  - Many Different Long-Running Operations May Change
- Each Interleaving Long-Running Operation May See Changes Since Last Time
  - Must Not Make Assumptions About Resources…
- Stateless Objects Provide Operations
  - “Reserve Hotel Room”, “Order Parts”, etc.
  - They Mask Changes
- Common Pattern
  - COM+ Components (formerly MTS)
  - EJB Session Beans

"Kinds of Data" Redux

Interesting!

	Request Response	Reference Data	Activity-Oriented	Resource-Oriented
Requires Open Schema for Interoperability	Yes	Yes	No	No
Encapsulation Useful	No	No	Yes	Yes
Immutable or Low Concurrent Update or Highly Concurrent Update	Immutable	Immutable (Versions)	Low Concurrent Update	Highly Concurrent Update
Fodder for Business Intelligence Analysis	Probably	Probably	Definitely	Definitely
Represent in XML?	Yes	Yes	Probably Not	Probably Not
Durable Storage in SQL?	XML Copy in SQL	XML Copy in SQL	Definitely	Definitely
Encapsulated Access?	No	No	Yes: Use Obj Persist	Yes: Stateless

May be copied using tools like XML → SQL stuff; shredding what you wa
to do BI Analysis on. May be transformed in and out using business log

The Ruling Triumvirate

SQL	It is fantastic to compare anything to anything and combine anything with anything in relational (within the bounded databa
XML	It is possible to have independent definition of schema and data XML infosets. You can independently extend, too.
Components/ Objects	Provide encapsulation of data behind logic. Ensure enforcemen business rules. Eases composition of logic.

Strengths and Weaknesses	Arbitrary Queries	Independent Data Definition	Encapsulation (Controls Data)
SQL Bounded Schema	Outstanding	Impossible: Centralized Schem	Not via SQL Enforced by DBA
XML Unbounded Schema	Problematic: chema Inconsistency	Outstanding	Impossible: Open Schema
Objects Encapsulated Data	Impossible: Can't See the Data!	Impossible Can't See the Data	Outstanding

Each model's strength is simultaneously its weakness!

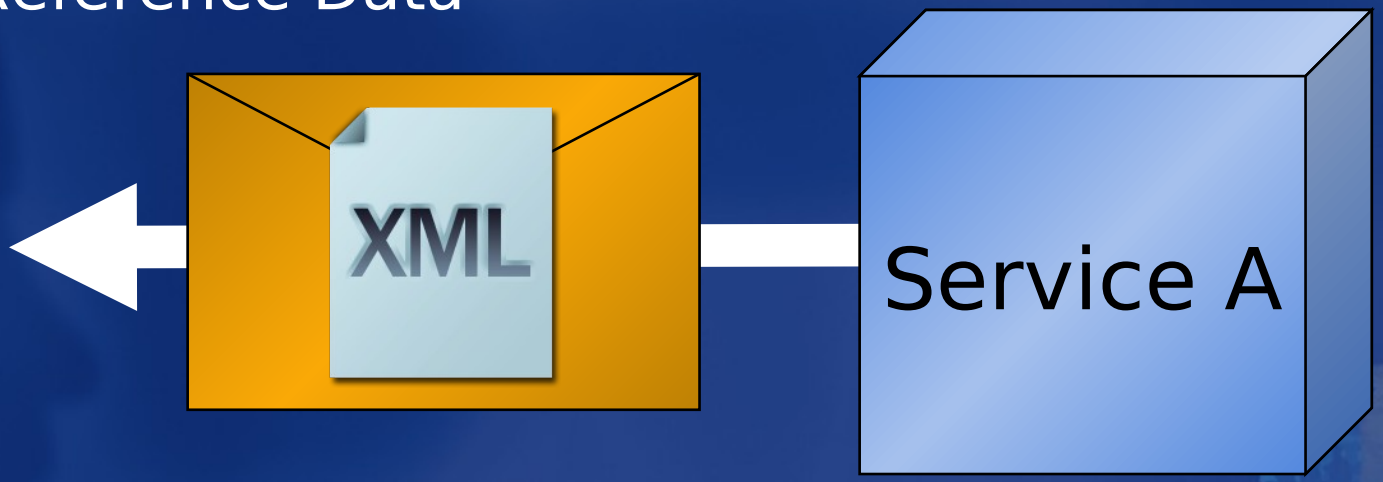
You can't enhance one to add features of the other without breaking it!

Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - **Considering Data and Messaging in Services**
    - The Space Between the Services
    - Schema in the Space Between the Services
    - Service Masters and Service Agents
    - Ownership of Data in the Enterprise
    - Representations of Data
    - **Tying It All Together**
  - Thoughts on Business Process

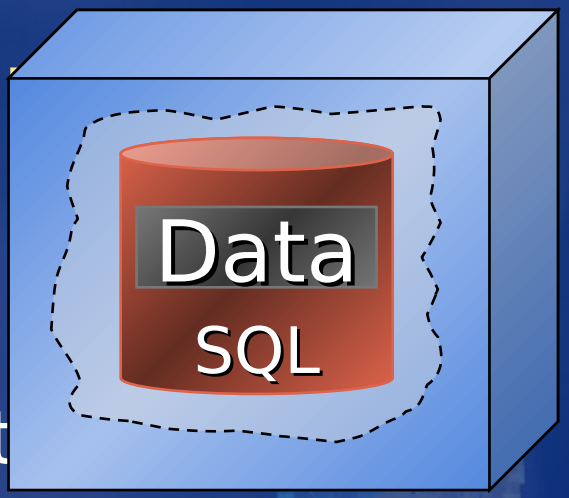
Talking to the World

- Use XML Infosets Across Services
  - Standard or Proprietary Marshaling Formats
  - Use XML Schema
- Interactions Across Services Include:
  - Requests,
  - Responses
  - Reference Data



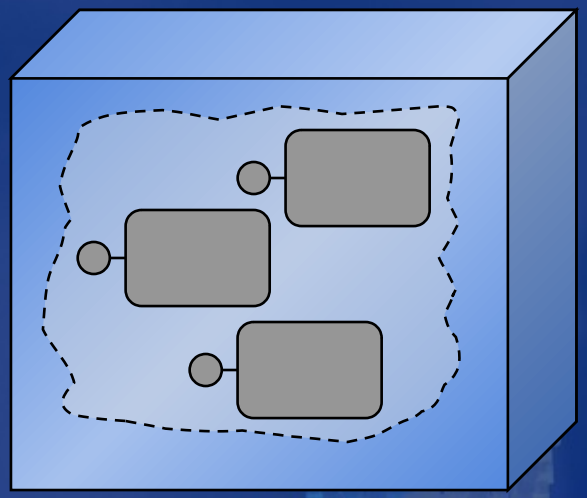
Storing the Data

- Business Services
  - Implement One or More Services
  - Encapsulate a Collection of Private Data
- Store Your Data in a SQL Database!
  - Industrial-Strength Store, Performance, and Management
- Store Incoming and Outgoing Messages in SQ
  - Remember Incoming and Outgoing for Auditing
  - Transactionally Record Outgoing Messages for Ret



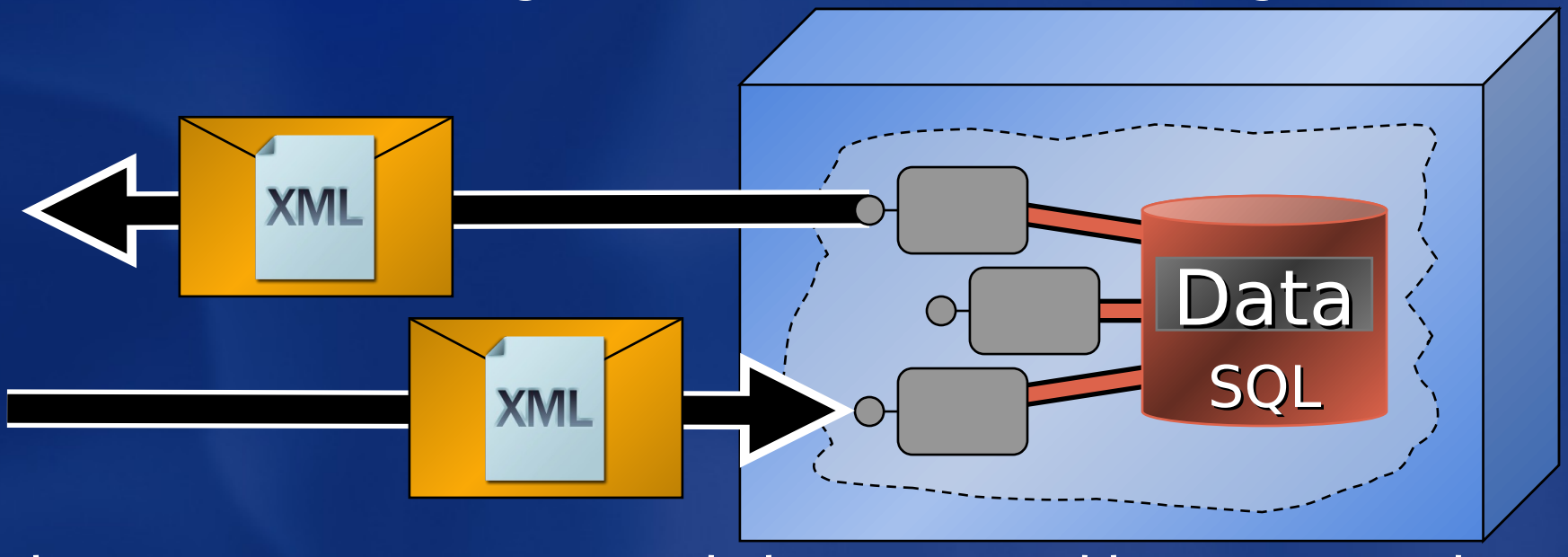
Implementing the Code

- Build Your Services with Objects
  - Offer Encapsulation of Parts Within the Service
  - They Are the Best for Software Engineering!
- Encapsulation
  - Business Services Are a Big Chunk
    - Completely Encapsulate Some Durable Data
  - Objects Make Littler Chunks
    - Inside the Business Service
    - Make It Easier to Build



Putting It All Together!

- Business Services Need All Three!
  - XML Infosets: Between the Services
  - Objects: Implementing the Business Logic
  - SQL: Storing Private Data and Messages



Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
  - Conclusion

Thoughts on Business Process

Interacting with Services	How can we reach agreements without transactions that span services?
Interacting with Masters and Agents	How do tentative operations get applied to Serv Masters? How about Service Agents?
The Power of Managing Uncertainty	Service Masters managed shared resources wit tentative work. This leads to uncertainty.
Schema, Contracts, and SLAs	It's all about the "black box" behavior of the service. How can we define this?
Wrapping Existing Apps with Biz Process	Humans make apps do long-running work. How can we make this driven by services more ofter
Layering for Business Process	We can build Service Agents just for managing biz process. This allows composable biz proces
Extrapolating Retail and Distributior	Looking at retail and distribution can tell us a lo about the composability of Biz Process.

Outline

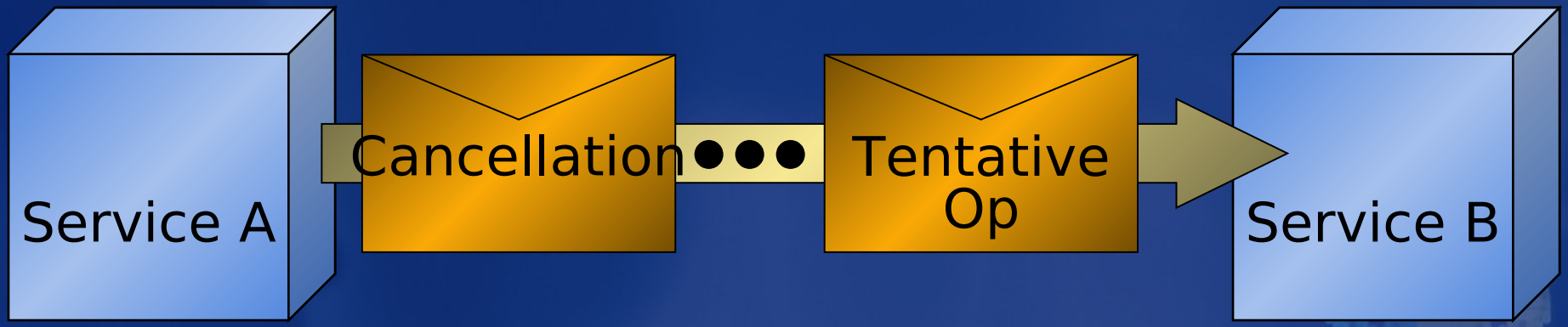
- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
    - **Interacting with Services**
    - Interacting with Service Masters and Service Agents
    - The Power of Managing Uncertainty
    - Schema, Contracts, and SLAs
    - Wrapping Existing Apps for Business Protocol
    - Layering for Business Process
    - Extrapolating Retail and Distribution
  - Conclusion

Contracts and Multiple-Message Interactions

- What About Multiple-Message Interactions?
  - Second Message Behavior Depends on First
  - Response in Request/Response Is an Example
- Have to Remember What Happened Before!
  - Associate New Message with Previous Messages
- Conversation
  - A Sequence of Messages Between Two Services
- Contract
  - The Allowable Sequences of Message Types

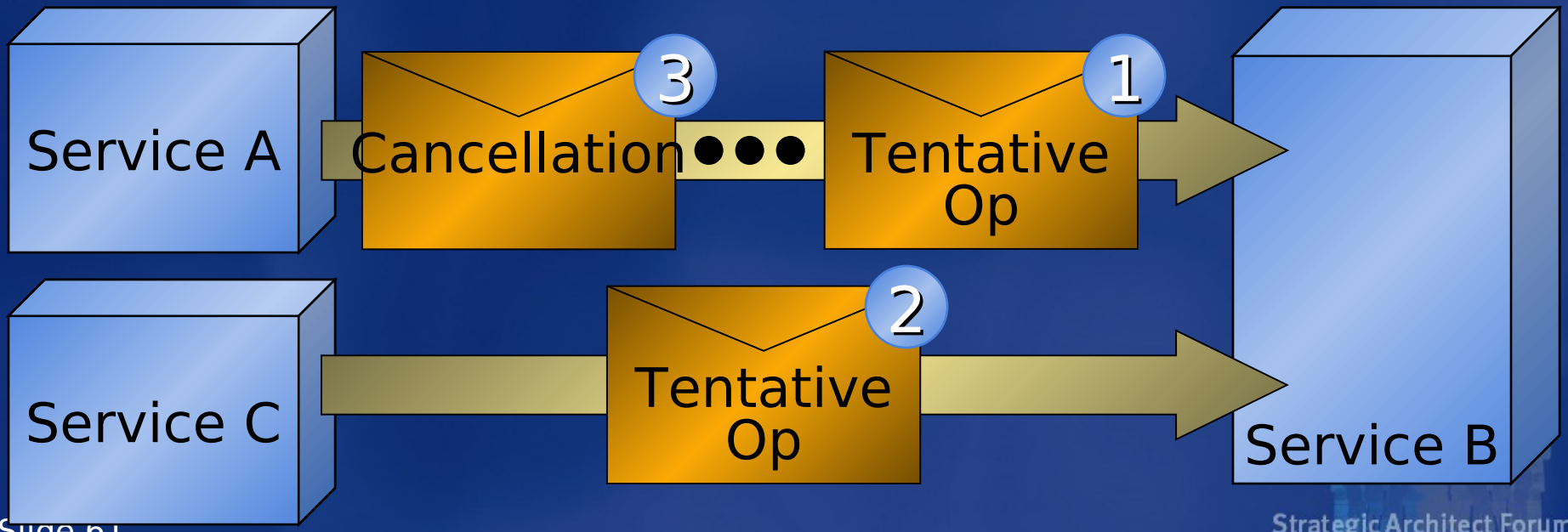
Tentative Operations

- Business Services Don't Share Transactions
  - This Is a Recommendation for Enterprise Usage Across Trust Boundaries
  - Now What Can We Do?
- Business Services May Accept Tentative Operations
  - Like a Reservation; May Be Cancelled Later
- If Cancelled, the Receiving Business Service Must Cope
  - Special Business Logic to Deal with Cancellations



Semantics of Tentative Operations

- Tentative Operations Must Be Reorderable
  - When Cancelled, a Compensation Must Occur
  - Other Operations May Have Occurred Since
- Operations and Cancellations Must Be Reorderable!



Semantics of Cancellation and Confirmation

- Cancellation
  - Cope with Not Doing Tentative Operation
    - Not Undo
    - New Operation to “Make Things Right”
  - Accepting Tentative Means It’s OK to Cancel
- Confirmation
  - Relinquish the Right to Cancel Tentative Op
  - Sometimes Time-Driven
    - Hotel Rooms Confirm in the Morning

Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
    - Interacting with Services
    - **Interacting with Service Masters and Service Agents**
    - The Power of Managing Uncertainty
    - Schema, Contracts, and SLAs
    - Wrapping Existing Apps for Business Protocol
    - Layering for Business Process
    - Extrapolating Retail and Distribution
  - Conclusion

Tentative Operations in Service Agents

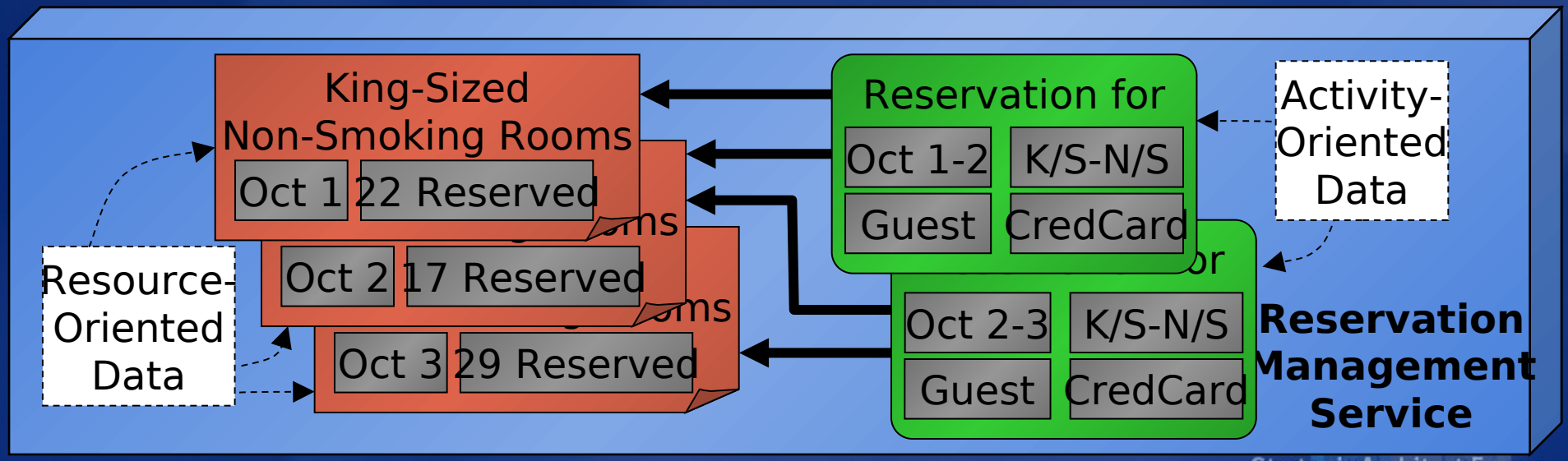
- Service Agents Encapsulate Activity Data
  - Activity-Oriented Data Is Used in a Single Long-Running Operation
  - Never Spans Long-Running Operations
- Tentative Operations Are Easy
  - Remember Commitments for Tentative Ops
  - Nothing Is Shared
    - Hard Problems Are in Resource-Oriented Data
    - That's Where the Sharing Is…

Travel Itinerary
- May Include Reserving Air Seats and Hot
- Ops with Airlines and Hotels for Reserva
- Travel Itinerary Works Indirectly
- Easy to Manage Tentative Changes

Shopping Basket
- Everything Is Tentative
- Trivial to Cancel

Tentative Operations in Service Masters

- Resource-Oriented Data
  - May Span Concurrent Long-Running Ops
- Activity-Oriented Data
  - Service Masters Make Commitments
  - Tentative Ops Are Commitments
  - Must Track Each Commitment with Activity-Oriented Data



Tentative Operations, Reorderability, and Interchangeability

- Long-Running Work Needs Tentative Ops
  - Allows Distributed Commitment
  - Invoked Service Accepts the Uncertainty
- Reorderability Is Essential for Tentative Ops
  - Easy with Activity-Oriented Data
    - Hold Reordering in the Activity-Oriented Data
  - Hard with Resource-Oriented Data
    - Stuff Shared Across Long-Running Work
- Interchangeability Provides Reorderability
  - One Resource Is As Good As Another

Outline

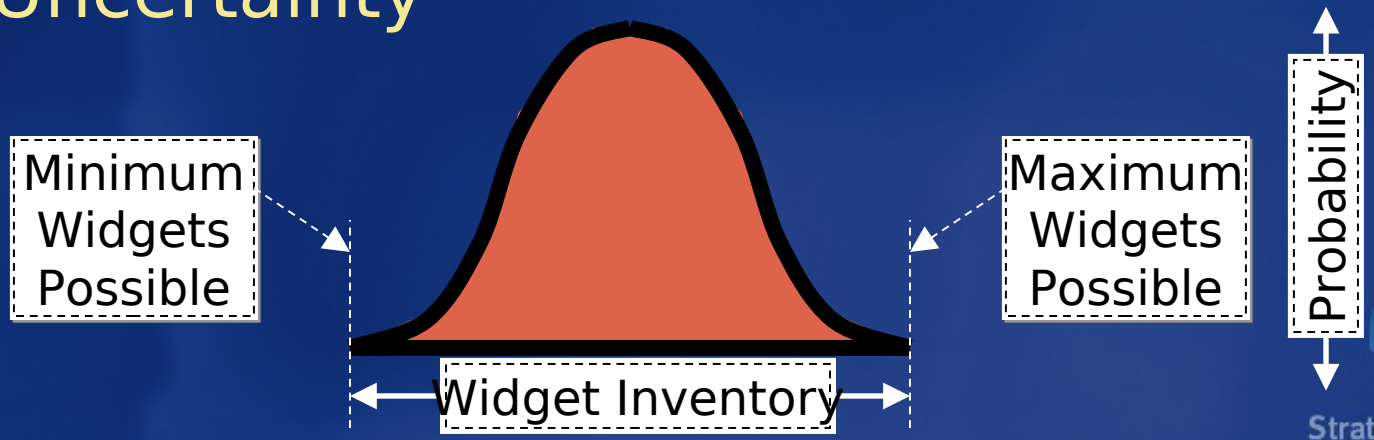
- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
    - Interacting with Services
    - Interacting with Service Masters and Service Agents
    - **The Power of Managing Uncertainty**
    - Schema, Contracts, and SLAs
    - Wrapping Existing Apps for Business Protocol
    - Layering for Business Process
    - Extrapolating Retail and Distribution
  - Conclusion

Increasing and Decreasing Uncertainty

- Tentative Operations Increase Uncertainty
  - You Get More Confused Accepting a Tentative Op
- Confirmations or Cancellations Decreases Uncertainty
  - Resolves Confusion from the Tentative Operation

Bounded Uncertainty

- Can Track the Worst-Case Values
  - Lowest and Highest Possible Can Be Tracked
- Tentative Moves Bounds Apart
  - Increasing Uncertainty
- Confirm/Cancel Moves Bounds Together
  - Decreasing Uncertainty
- Knowing Bounds Gives Bounded Uncertainty



Acting on Bounded Uncertainty

- Knowing Bounds Allows Different Business Rules:
  - Refuse Order That May Overflow Warehouse
  - Calculate Cost of Temporary Storage vs. Value of Accepting the Order
  - Order Hotel Food Based on Reservations and Probabilities
- Interesting Potential for Risk Management
  - Just Like Futures and Commodities Calculations

Outline

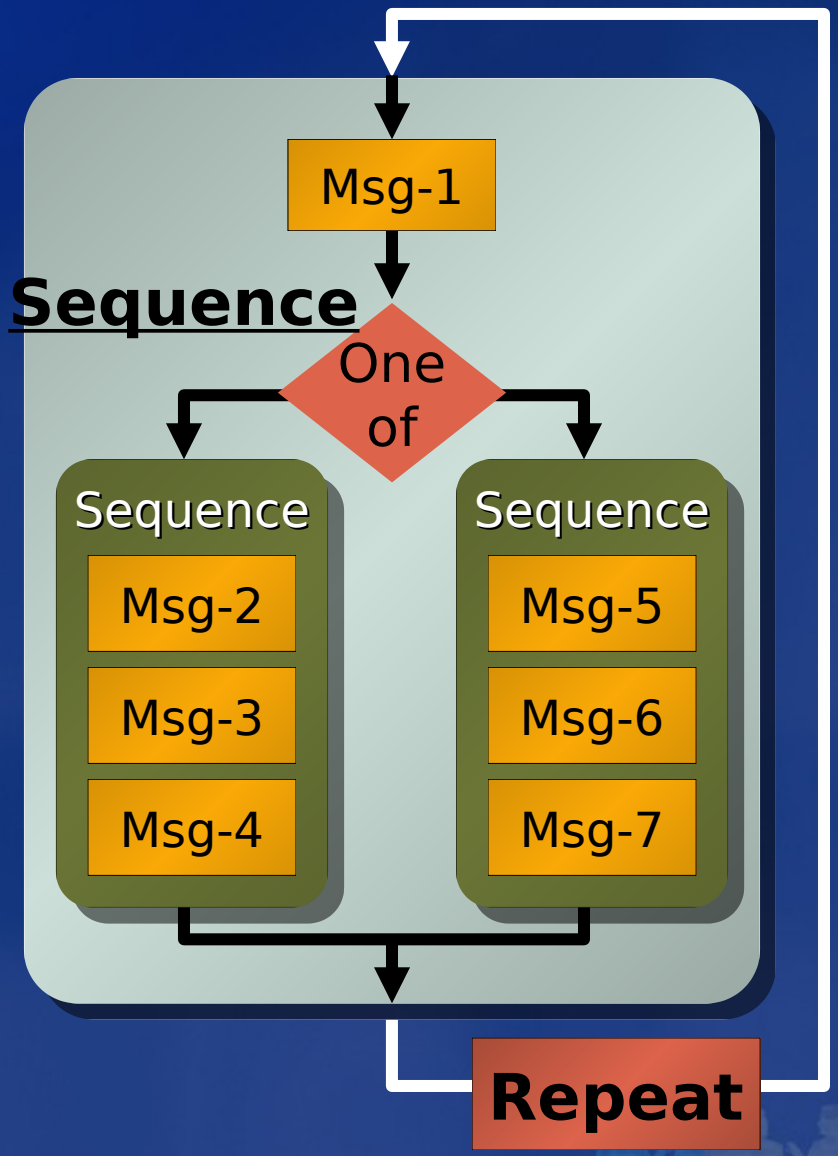
- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
    - Interacting with Service Masters and Service Agents
    - The Power of Managing Uncertainty
    - **Schema, Contracts, and SLAs**
    - Wrapping Existing Apps for Business Protocol
    - Layering for Business Process
    - Interacting with Services
    - Extrapolating Retail and Distribution
  - Conclusion

Messages and Schema

- Message Schema Must Be Shared
  - Reference Data Published by Target
- Ongoing Industry Evolution
  - XSD (XML Schema)
    - Defines XML Infoset Schem
  - WSDL (Web Services Description Language)
    - Defines Parameter Marshaling
- Loose Coupling and Late Binding
  - More Doc-Centric; Less Method Call–Centric
- No Clear Consensus on Formats Yet

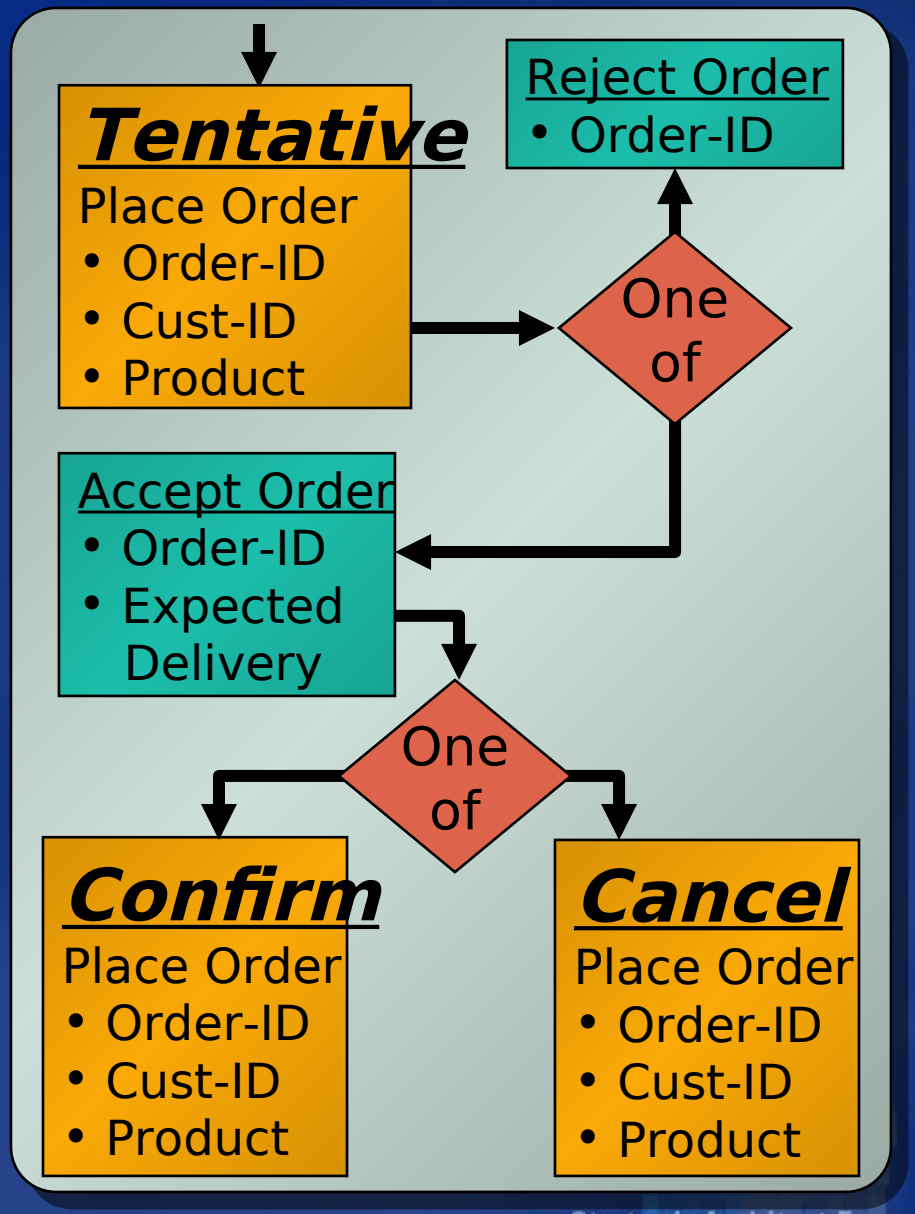
Contracts

- Define Allowed Message Sequencing
  - You Send PO, I'll Send Ack or Reject, etc.
- Ongoing but Nascent
  - Active Area of Research
  - Cool Mappings to Formal Languages
  - Cool Possibilities for Tying to Orchestration



Contracts and Tentative Operations

- Annotate Tentative, Confirm, and Cancel
  - Contract Flow Defines Outcome of Operations
- "Black Box" Definition of Uncertainty
  - Which Partner Will Resolve Uncertainty?
- Message Flow for Resolving Uncertainty



Source-to-Target Message

Target-to-Source Message

Service Level Agreements: A Gleam in Our Eye

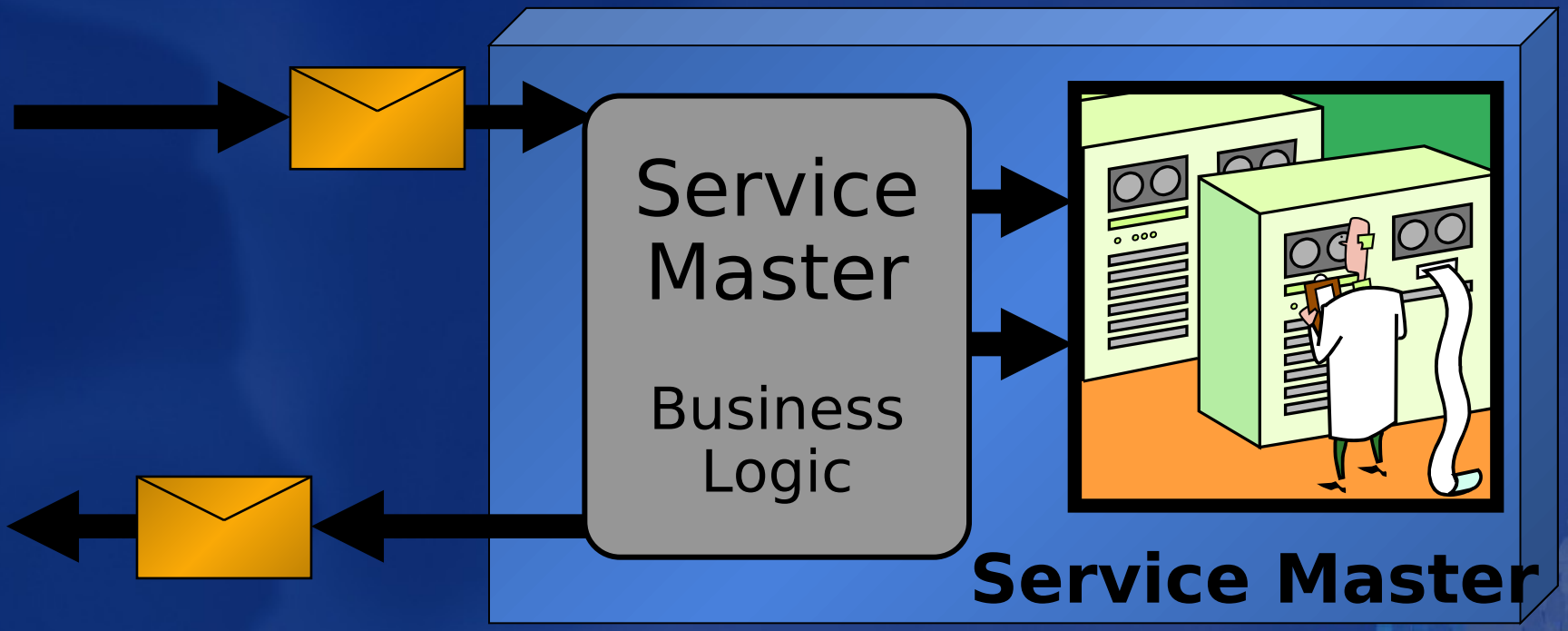
- Service Level Agreements
  - Performance Guarantees
  - Timeliness
  - Business Compensation for Failure to Perform
  - Quality Definitions
- Not Even Nascent…
  - Formalizing This Is Still a Fond and Distant Hope
- Time-Outs
  - Essential Tool for Message Interaction
  - Best We Have Today

Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
    - Interacting with Services
    - Interacting with Service Masters and Service Agents
    - The Power of Managing Uncertainty
    - Schema, Contracts, and SLAs
    - **Wrapping Existing Apps for Business Protocol**
    - Layering for Business Process
    - Extrapolating Retail and Distribution
  - Conclusion

Making a Service Master by Wrapping an Existing App

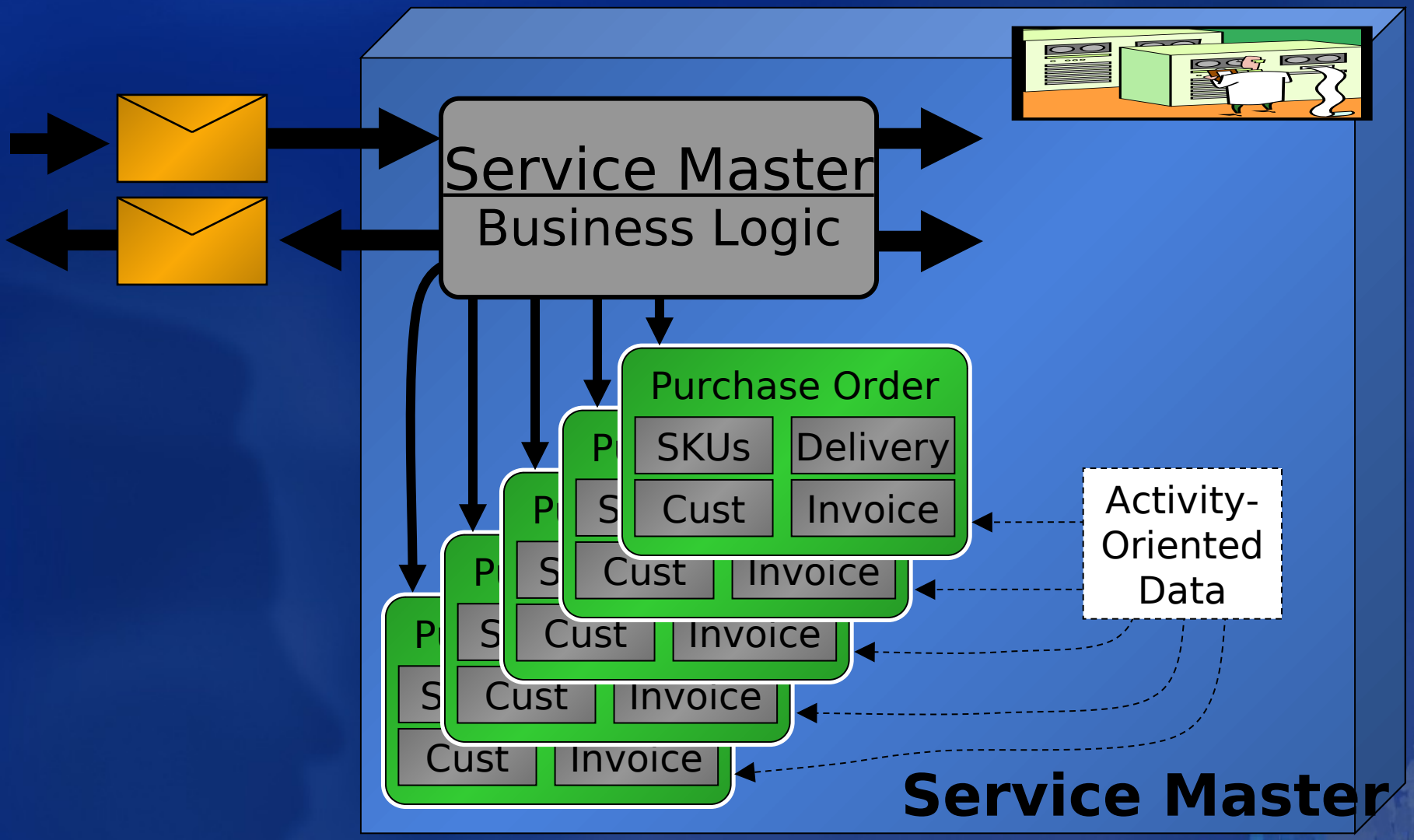
- We Can Wrap an Existing Application
  - Since It Manages Resources, It's a Service Master



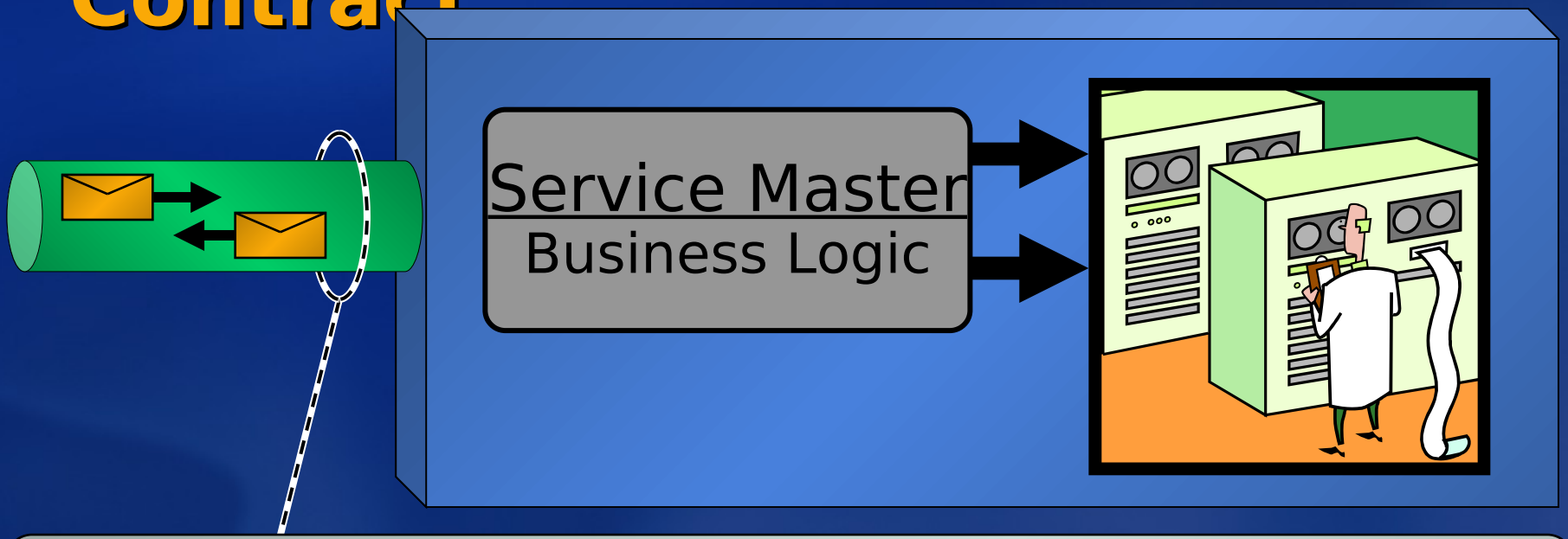
Identifying Cancelable Business Operations

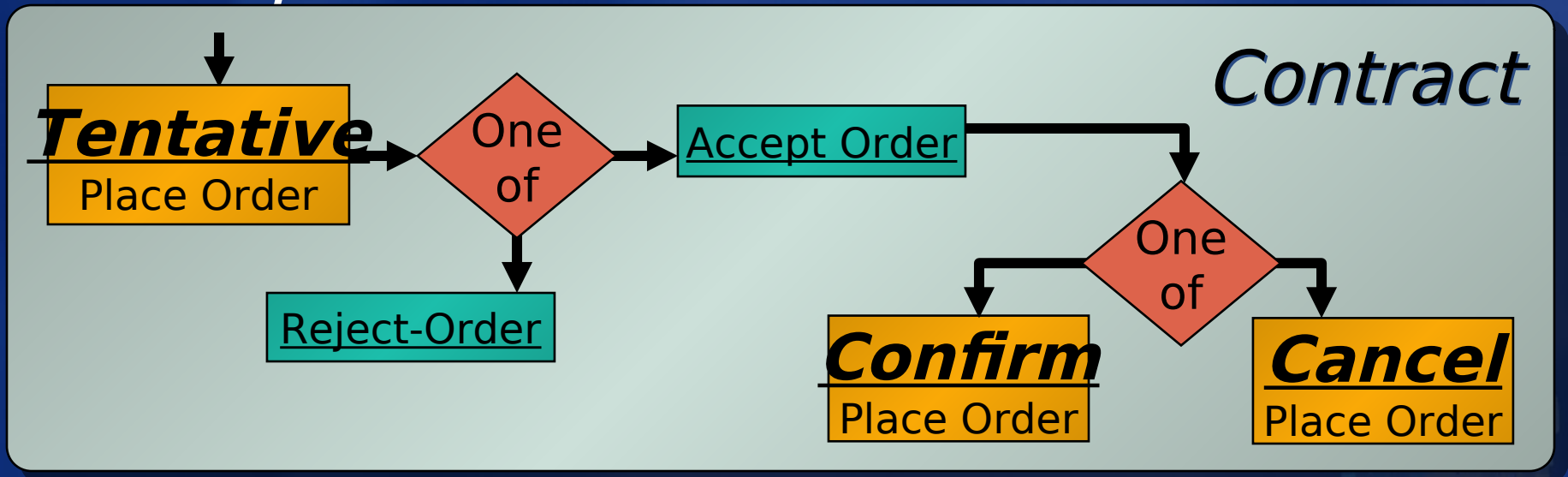
- Analyze the App to Identify Its Operations
  - Humans Perform Operations with the App
  - Many of These Are Cancelable
    - The Cancellation May Take Many Steps
- The Goal Is to Capture “Low-Hanging Fruit”
  - Identify Easy-to-Automate Human Interactions
  - Wrap Those As Services
- If Too Hard to Automate, Enqueue for Humans
  - Ensure the Requests Aren’t Lost
  - OK to Get Human Help
- Try to Automate Cancellation and Confirmation

Remembering Obligations in the Wrapping Service Agent



Exposing a Service Master Contract



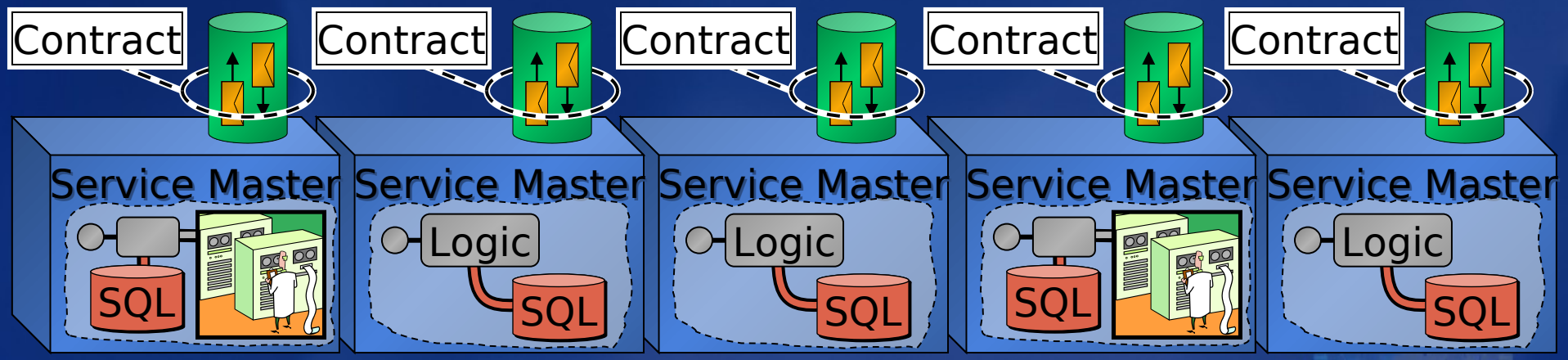


Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
    - Interacting with Services
    - Interacting with Service Masters and Service Agents
    - The Power of Managing Uncertainty
    - Schema, Contracts, and SLAs
    - Wrapping Existing Apps for Business Protocol
    - **Layering for Business Process**
    - Extrapolating Retail and Distribution
  - Conclusion

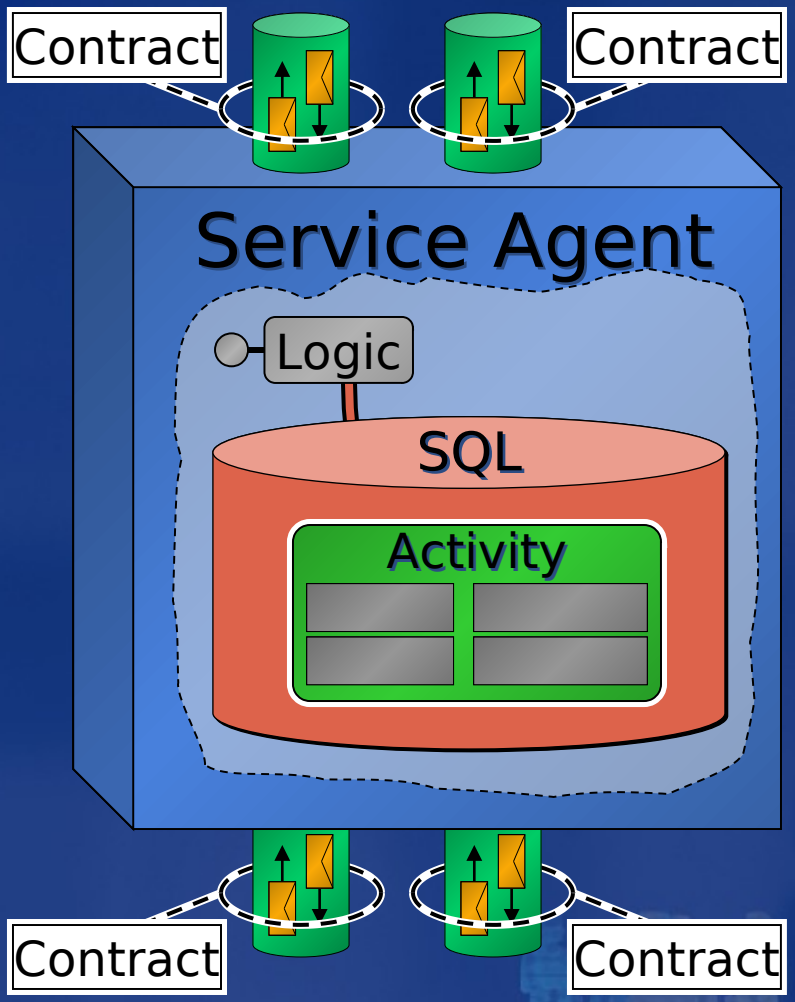
Service Masters Managing Resources

- Service Masters Manage Resources
  - Sometimes Embedded in Legacy Apps
  - Sometimes Managed by the Service Code Itself
- Service Master Export Contracts
  - These Support Tentative, Cancel, and Confirm Ops

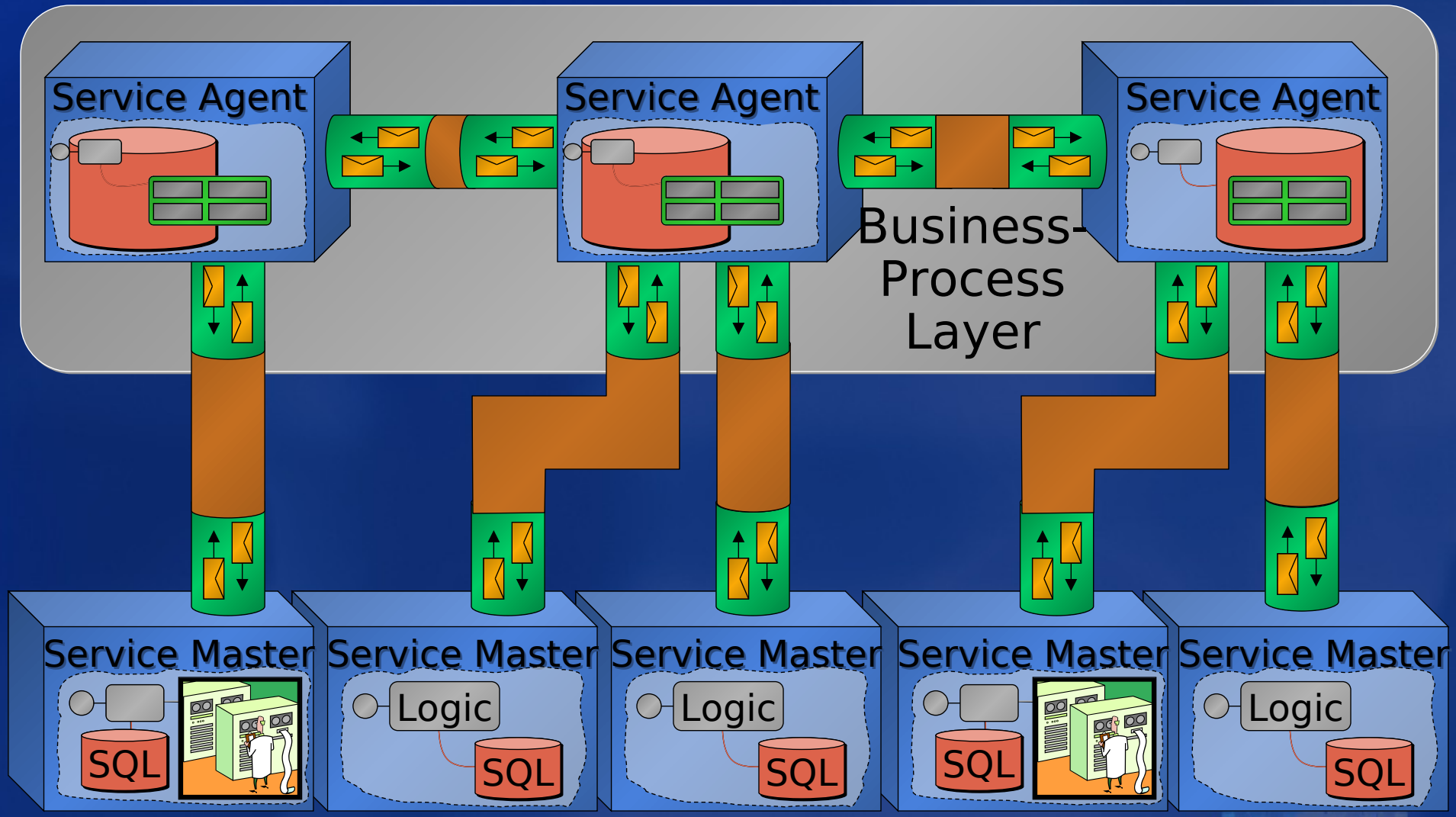


Dedicating Service Agents to Business Process

- Service Agents Can Do Business Process
  - State of the Business Process Kept in Activity Data
- May Have Many Contracts
  - Based on the Business Process
  - One Contract Starts the Service Agent
  - May Have Many Contracts for Outgoing Work



Composability of Business Process Agents

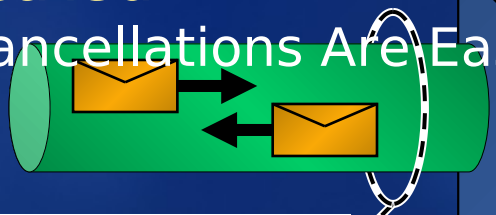


Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - **Thoughts on Business Process**
    - Interacting with Services
    - Interacting with Service Masters and Service Agents
    - The Power of Managing Uncertainty
    - Schema, Contracts, and SLAs
    - Wrapping Existing Apps for Business Protocol
    - Layering for Business Process
    - **Extrapolating Retail and Distribution**
  - Conclusion

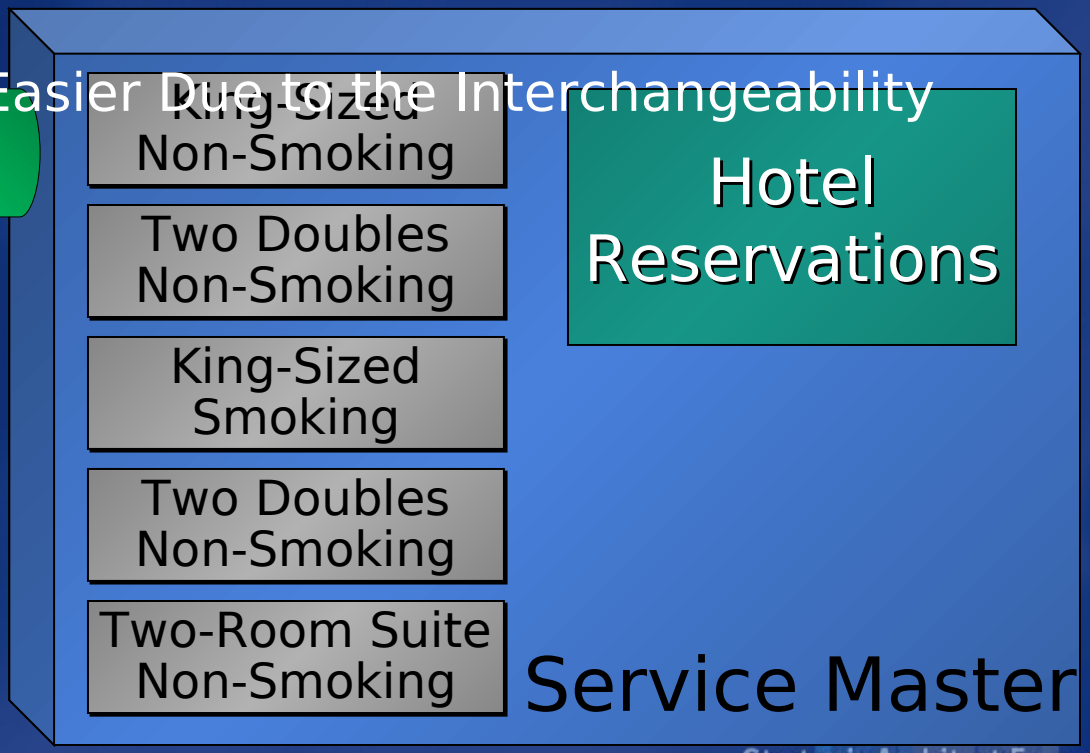
Interchangeability and Service Masters

- Service Masters Should Offer Interchangeable Resources
  - Putting Resources into Buckets
  - Allowing Allocation of Resources That Fit into These Buckets
- An Interchangeable Resource Can Be Tentatively Allocated
  - Cancellations Are Easier Due to the Interchangeability



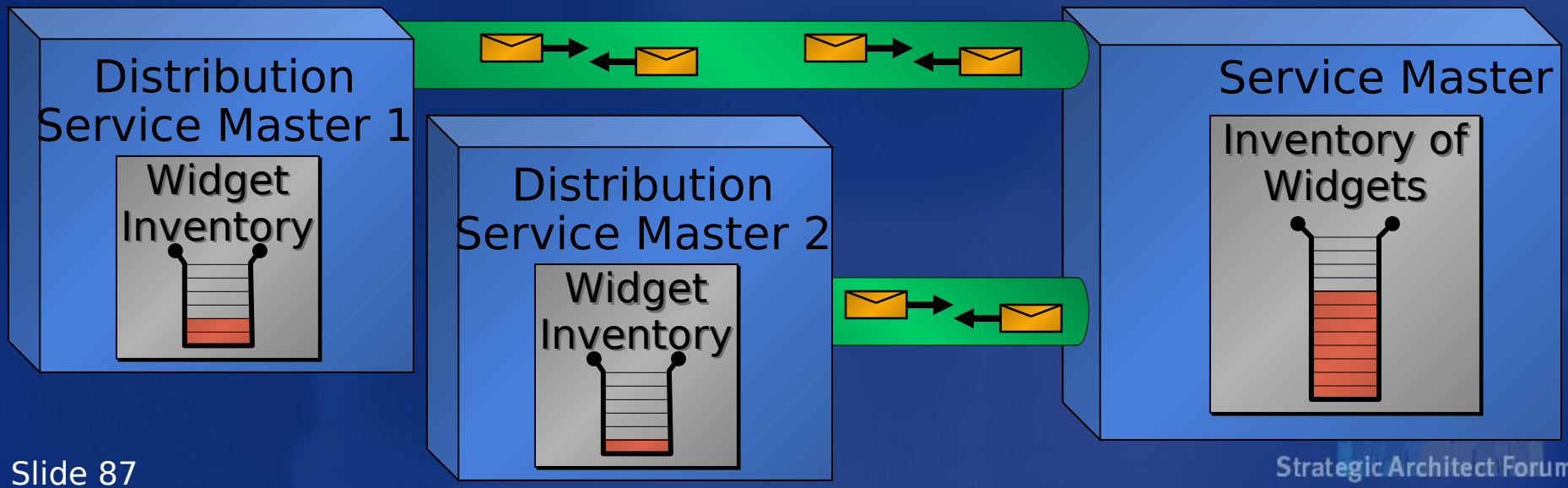
5 Categories of Rooms

- Requests Accepted for Only These Categories
- Tentative Requests Accepted
- Blocks of Rooms Available



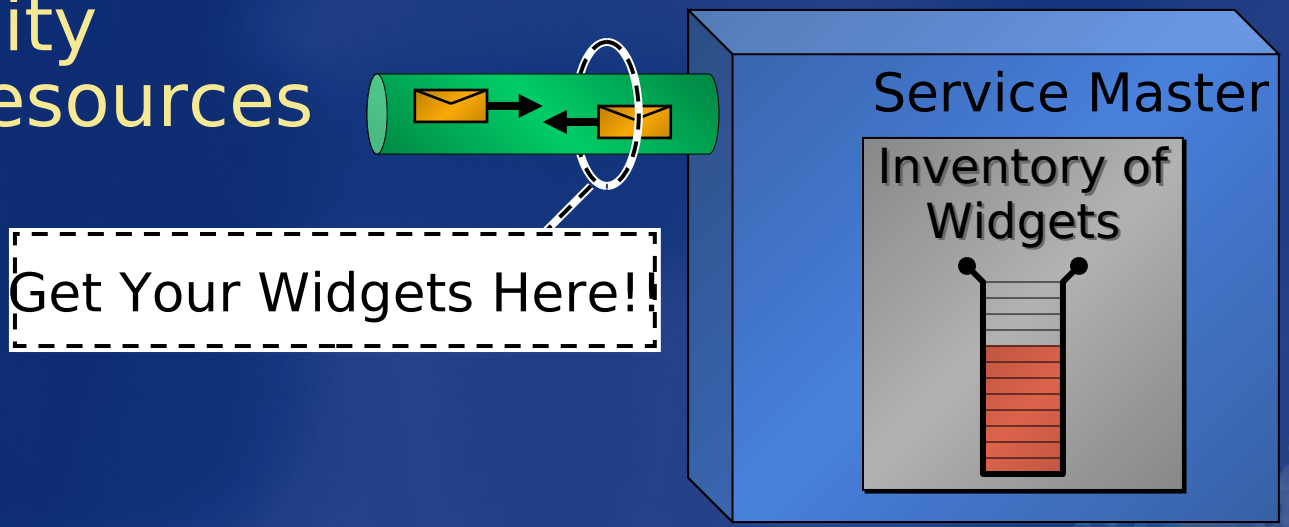
Filling a Distribution Channel

- Interchangeable Resources Can Be Commodities
  - One Is Just As Good As Another
- You Can Pre-Allocate Interchangeable Resources
  - The Service Master Likely Has a Supply of Them
- A Distribution Channel Can Be Made
  - Just Like Physical Goods, You Can Pre-Allocate Them
  - Allocated Resources Can Be Passed to Another Service



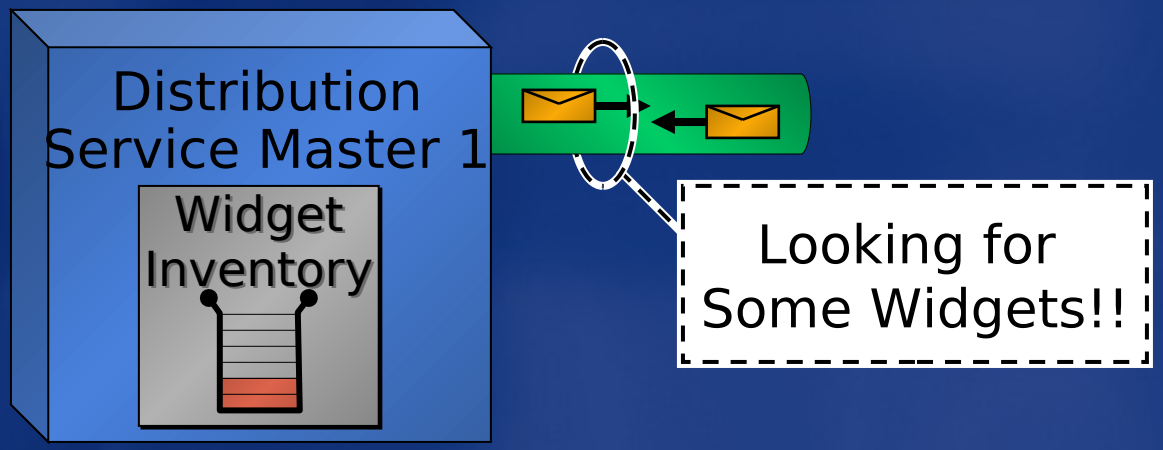
Exposing the Supply of Interchangeable Resources

- A Service May Offer a Supply of Interchangeable Resources
  - It Is Ready for Someone to Come and Consume Those Resources
- One Challenge Is to Advertise the Availability of the Resources



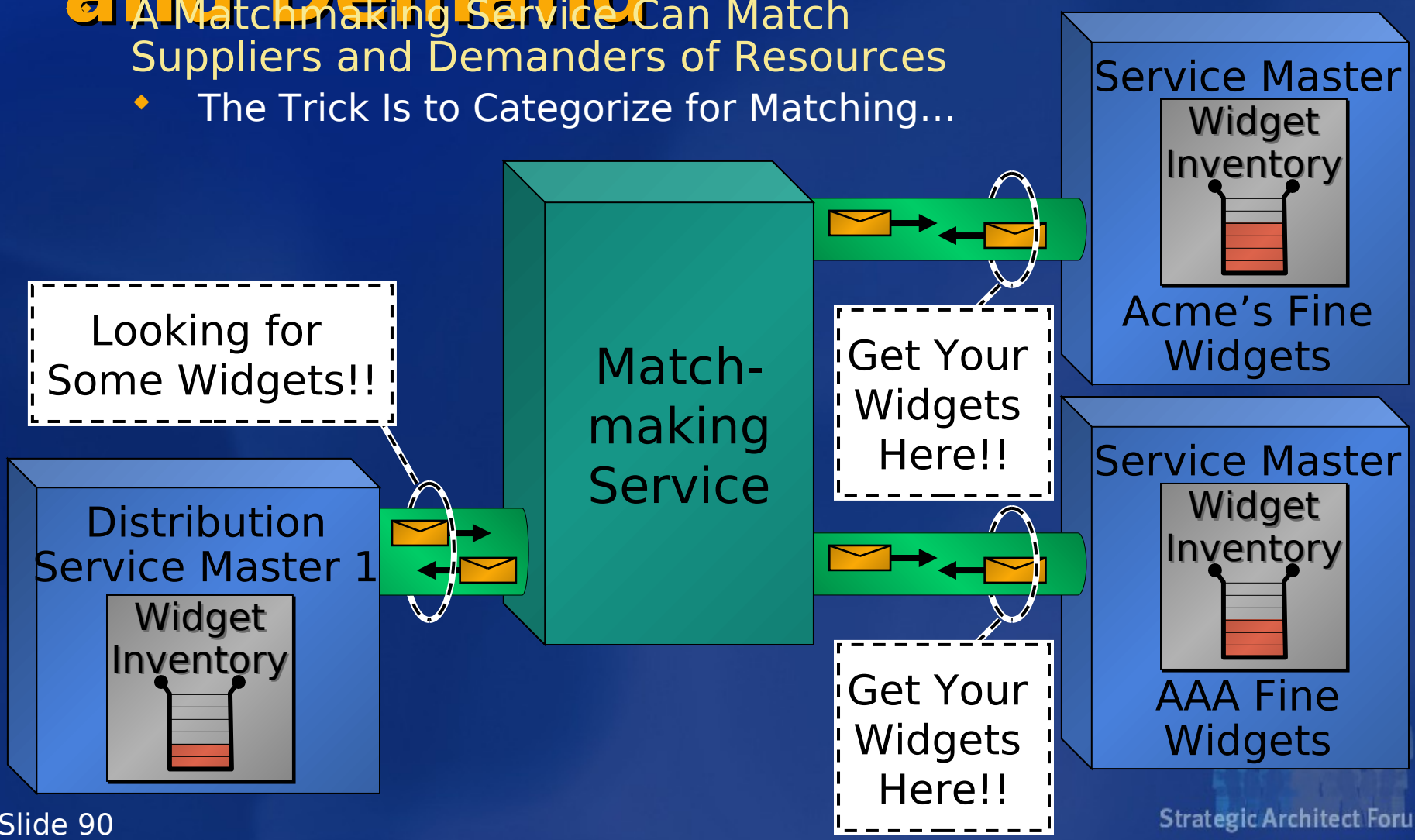
Exposing the Demand for Interchangeable Resources

- A Service May Issue a Demand for Interchangeable Resources
  - It Wants Stuff that Meets Certain Requirements



Brokering Matches of Supply and Demand

- A Matchmaking Service Can Match Suppliers and Demanders of Resources
  - The Trick Is to Categorize for Matching…



Outline

- Metropolis: Part 1
  - Metropolis: The Analogy
  - Practical Advice for Building Services
  - Concluding Part 1
- Metropolis: Part 2
  - Introduction to Part 2
  - Considering Data and Messaging in Services
  - Thoughts on Business Process
  - **Conclusion**

Eli Whitney Was Right!

- Interchangeability Is the Key
  - Making Pools of Equivalent Resources
  - Allows a Marketplace of Resources
    - Efficient Consumption of Resources
    - Efficient Production of Resources
- Need Standards for Interchangeable Stuff
  - De-Facto Standards Emerge by Marketplace Leadership
- Ambiguity Increases Interchangeability
  - Specify Only What Matters
  - Extensions Specify Details

Supply: Hotel Room
- General Info:
  - Mid-Town; Times Square
  - 3-Star
  - Vacancies Thurs, Fri, Sat, Mon
- Details: ...

Request: Hotel Room
- Need:
  - Friday Night 11/7/03
  - Mid-Town Manhattan
  - Not a Dump
- Want
  - Exercise Facility

Rambling Philosophy…

- Atomic Transactions Are Singularities
  - Locking → A Single Point in Time
  - Two-Phase Commit → A Single Point in Space
- Hard to Spread Work Across Space and Time
  - Different Services → Across Space
  - Different Messages → Across Time
- Reorderability Relaxes Space and Time
  - Gives Acceptable Answers
  - Interchangeability Empowers Reorderability
- Service Agents Don't Mind Reordering
  - Use Read-Only and Activity-Oriented Data
  - Service Masters Have a Harder Job with Reorderability

Implications of Metropolis

- Heterogeneity Happens!
- Ongoing IT Investment
  - Infrastructure vs. Business
  - Historic Monuments
- Standardization Is Nascent
  - Connection Largely by People
  - Efficiencies Still to Come
- Business Process Is Nascent
  - Still Mostly Ad-hoc
  - Growing to Become Dominant Force
- Loose Coupling Helps Investments

Envisioning the Service-Oriented Enterprise

- Services Move Us Forward
  - Lots of Islands of App Code
  - Services Are for Connecting the Islands!
- Independence Is Essential
  - Services Evolve Independently
  - Build vs. Buy
- Inside Your Company and Across Companies
  - Connecting Your Disparate Apps Comes First!
  - Hooking to Partners Is Important, Too!
- Services Cleave Your Applications
  - They Cleave Them Apart into Independent Pieces
  - They Cleave Them Together Allowing Interaction

Build Your Services Now!

- Microsoft Is Investing Lots in Services
  - Web Services Define Interop Standards
  - “Longhorn” Makes Service Development Easier
- You Can Build Services Now!
  - It’ll Be Easier with “Longhorn” but Don’t Wait!
  - Your Business Needs Services
  - Guidance Available to Help
    - http://msdn.microsoft.com/architecture
- Invest Now in Building Services!
  - Separate and Connect Your Apps As

Questions?

Microsoft®